Query Space Reduction in Information Retrieval

by

Fergus Kelledy M.Sc.

A Dissertation Presented in Fulfilment of the Requirements for the Ph.D. Degree

Supervisor: Dr. Alan F. Smeaton

School of Computer Applications

February 1997

Declaration.

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy in Computer Applications, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my own work.

Signed: Fergus Kelledy Date: 6/2/97

Fergus Kelledy.

I would like to dedicate this thesis to the memory of my Grandmother

Kathleen McComish.

"Simply the best..."

Acknowledgements.

Well its been a long time coming but I guess I'll finally have to leave DCU. There are of course a number of people to whom I owe a great deal.

Firstly, I would like to thank my supervisor Dr. Alan Smeaton, whose assistance, guidance and unflagging enthusiasm in my research proved invaluable. Special thanks and appreciation must go to my mother for her good advice, constant support and encouragement without which I would not be where I am today.

Special mention must go to Carmel for making the later half of my stay in the lab a lot more enjoyable than the first! Thanks must also go the members of the MMIR group namely Gavin, Fran, Mark and Ger for their uncanny knack in finding bugs in every piece of code I wrote.

Lastly I would like to thank all other members of the Lab (past and present) for long coffee breaks, even longer lunch breaks and also for making the lab an interesting place to work for the past number of years.

Fergus Kelledy,

February 1997.

Abstract.

Today's rapidly expanding and dynamic information age coupled with users who are becoming more discerning about what information they want and when they want it poses a serious challenge to information retrieval systems in their attempt to match user's information needs with information repositories.

To date most research on information retrieval has concentrated on improving system effectiveness. However as the amount of online information and the number of users concurrently accessing this information continues to grow at an exponential rate the efficiency of information retrieval systems is now a core concern of information retrieval system developers. Users who were previously content to wait for information they needed are no longer willing or able to do so because in today's dynamic information age the 'shelf life' of information is getting shorter and shorter. This results in increasing pressure on information systems to provide the 'right' information at the 'right' time.

This research focuses on the improving the efficiency of information retrieval systems. To this end we have developed and implemented a number of techniques aimed at reducing system response time by reducing the amount of data processed in order to effectively respond to a user's information need.

Table of Contents.

Table of Figures.

Table of Tables.

1. Introduction to Information Retrieval.

1.1 Overview.

This Chapter will, firstly, define the context within which we are working by giving a brief history of the origins of information science followed by a more detailed definition of the term 'Information Retrieval'. Secondly, we describe the problem being addressed. Thirdly, we explain the need to address this problem and, lastly, we outline the approaches that will be taken to solve the problem.

1.2 Origins of Information Retrieval.

The roots of information retrieval are in documentation, a field that emerged when digital computers were developed during the 1940s and early '50s. During World War II the need arose to increase the precision and depth of bibliographic searches, resulting in efforts to change traditional kinds of classification into computer-compatible systems. Automated searching of files, co-ordinate indexing, and controlled vocabularies were introduced in response to the urgent need to create easy access to the contents of scientific journals. Automated abstracts, or summaries, of documents were then developed to further simplify access to research findings. In the 1960s massive collections of documents were transferred to databases or converted to non-print forms; various searches could then be done by computer. By 1980 information science had become a thoroughly interdisciplinary field.

Since the 1940's problems associated with information storage and retrieval have attracted ever more interest. The essence of the problem is the ever increasing amount of information available to us to which accurate and speedy access is becoming ever more difficult. The net result of this is that relevant information is ignored since it is never found. This in turn leads to a great deal of duplication of work and effort. Since the advent of computers, a great deal of thought has been directed towards using them to provide rapid and intelligent retrieval systems. Computers have been successfully incorporated into certain aspects of the information storage and retrieval problems and some of the more tedious tasks, such as cataloguing and general administration, have

successfully embraced the use of computers. However the problem of effective / intelligent retrieval on a large scale remains for the most part unsolved.

1.3 Formal Definition of Information Retrieval.

'Information retrieval' (IR) is an academic discipline and also an industry that deals with the generation, collection, organisation, storage, retrieval, and dissemination of recorded knowledge. IR is a wide and often loosely-defined concept and as a result of this certain qualifications need to be applied in order to more accurately define what we are taking about. Figure 1.1 gives us the formal dictionary and thesaurus definitions of the words 'Information' and 'Retrieval'.

Within our context, **'in-for-mat-ion**[3]**'**: *'A collection of facts or data'*, and **'re-trieve**[2]**'**: *'To find and carry back; fetch'*, are the most appropriate. To this end our formal definition of 'Information Retrieval' is *'Finding and bringing back relevant items from a collection of facts or data in response to a request'*.

Dictionary	
in-for-ma-tion	re-trieve
noun. Knowledge derived from study or experience. Knowledge of an event or situation; intelligence. A collections of facts or data. Informing or being informed; communication of knowledge. in'for-ma'tion-al *adj.*	*Verb.* To get or bring back; regain. To find and carry back; fetch. re-triev'a-ble *adj.* re-triev'al *noun.*

Thesaurus	
information	retrieval
noun. An account of current events. *Syn.:* News, Intelligence, Tidings, Word, News flash, Scoop, Bulletin, Communiqué, Announcement, Report, Release, Dispatch, Article, Piece, Account.	*noun.* The act of getting back or regaining. *Syn.:* Restoration, Reclamation, Recouping, Recovery, Redemption, Repossession, Rescue, Salvage.

Figure 1.1 Formal Definition of 'Information Retrieval'.

To be even more specific we are dealing with 'automatic' IR. Automatic as opposed to manual and information as opposed to data or fact. Figure 1.2 clearly illustrates some of the major differences between IR and data retrieval (DR) [van Rijsbergen 1979].

	Data Retrieval (DR)	Information Retrieval (IR)
Matching	Exact Match	Partial match, best match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Classification	Monothetic	Polythetic
Query Language	Artificial	Natural
Query Specification	Complete	Incomplete
Items wanted	Matching	Relevant
Error response	Sensitive	Insensitive

Figure 1.2 Data Retrieval Vs Information Retrieval.

Going into more detail on each item in Figure 1.2 we have exact matching in DR. This involves checking to see whether or not an item or a record is present in a file. In IR we have partial matching which is finding those items that partially match the request and then selecting the best subset of those items in response to the request. Deductive inferencing is used in DR, that is, if A implies B and B implies C then A implies C. In IR it is far more common to use inductive inference, with relations specified only by a degree of certainty or uncertainty, hence our confidence in the inference is variable. This distinction results in describing DR as deterministic and IR as probabilistic.

DR is more likely to be interested in a Monothetic classification i.e. one with classes defined by objects possessing attributes both necessary and sufficient to belong to a class. Within the IR field such a classification is not very useful but Polythetic classifications are i.e. individuals within a class will possess only a proportion of all the attributes possessed by all members of that class. In such cases, no single attribute is necessary or sufficient for membership of a class. The DR query language will be artificial and generally complete in nature, with a restricted syntax and vocabulary while the IR query language will be natural and invariably incomplete. In IR we are searching for relevant items as opposed to exactly matching items in DR. The extent of

the match in IR gives some indication of the likelihood of relevance of that item to the request. A consequence of this is that DR is more sensitive to errors i.e. an error in DR retrieval implies total failure of the system, while in IR, small errors do not significantly affect system performance.

1.4 Applications for IR Systems.

In recent years the increased availability of media be it the texts of books, newspapers, magazines, etc., in a machine readable format (via the World Wide Web for example) has meant that people need access to this information. Since this information is typically either unstructured or loosely structured it is not suitable to be managed by DR systems which require their information sources to be structured in nature. This is where IR systems step in and take over from the more conventional DR systems. IR systems must firstly cope with the unstructured and highly variable nature of the information they are dealing with and secondly, match users information needs as best they can against the available information collections they have control over. It is within this area that IR systems are coming into their own. This emergence has been more notable since the Internet, particularly the World Wide Web has become popular. People now have easy access to vast quantities of on-line information to which IR systems must facilitate the delivery of what the users want when they want it.

1.5 Components of IR Systems.

Figure 1.3 simply illustrates the overall construction of a typical automatic IR system. The illustration consists of three major components namely, the input, the processor and the output. Initially this may seem a little over simplistic but it is an ideal place from which to start.

Starting with the first sub-component of an automatic IR system, the input, the main problem here is to obtain a representation[1] of all inputs to the processor in a form

---

[1] The process by which representation is generated will be outlined in greater detail in the next section.

which is suitable for a computer to use. It must be stated that computer-based IR systems only store a representation of their inputs (documents and queries). For example an internal document representation could simply be a list of extracted words deemed to be significant. It is also possible to modify the initial input query via a process called relevance feedback where information is fed back into the system by the user in response to the system's initial output in an effort to improve the results of subsequent retrieval.

The next sub-component of an automatic IR system is the processor. The processor is concerned with manipulating the internal query and document representations so as to achieve meaningful and effective results while being as efficient as possible in doing so. The last sub-component is the output. This usually consists of a set of document identifiers ranked in order of relevance to the given input query. These document identifiers can subsequently be used to allow the user to view and make relevance judgements on the documents presented.

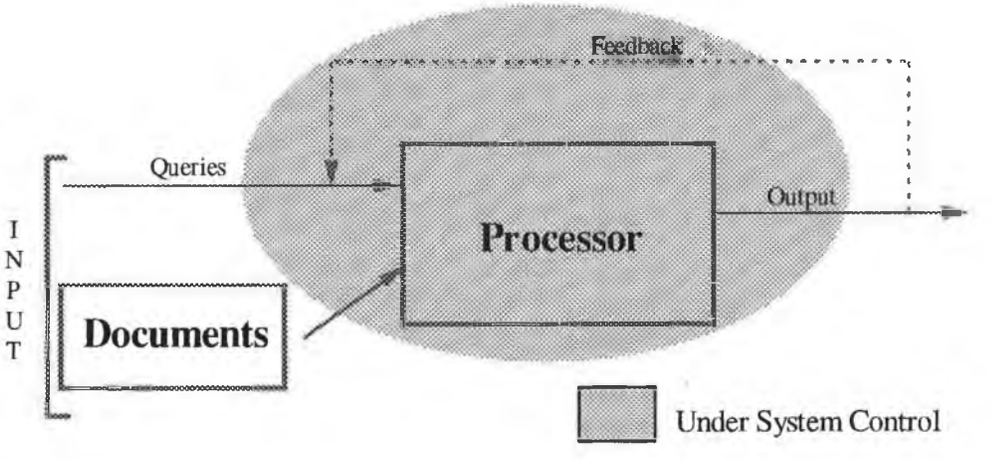


Figure 1.3 Structure of a typical IR System.

1.6 Automatic Text Analysis.

In order for an automatic IR system to actually operate on given information, the information must be stored within the computer in some internal representation. It is very unlikely that this internal representation of the information will correspond to

the original form of the documents[2] in that certain aspects of the formatting and structure of the original information will be removed and other additional information will be incorporated into the internal representation. The process for generating this internal representation can be loosely defined as automatic text analysis. There are a number of approaches to automatic text analysis which vary from statistical to linguistic methods. Linguistic text analysis, a very large area in itself, can be further broken down into syntactic (structure of text) and semantic (meaning of text) analysis. In general, linguistic analysis (syntactic, semantic and pragmatic) has proved expensive to implement and it is still not clear how the result of such linguistic analysis could be used to enhance an IR system. Part of the above problem is that relatively little progress has been made on developing a formal semantic theory. Such a theory, if developed, would have great and far reaching consequences for the development of intelligent IR systems.

A formal semantic theory is not a pre-requisite for a good IR system. The statistical approach[3] has been found to be moderately successful. In [Luhn 1958] he states: *'It is here proposed that the frequency of word occurrence in an article furnishes a useful measurement of word significance. It is further proposed that the relative position within a sentence of words having given values of significance furnish a useful measurement for determining the significance of sentences. The significance factor of a sentence will therefore be based on a combination of these two measurements.'* In summary, his assumption means that frequency information can be used to extract words and sentences from within a document i.e. its internal representation. Let f be the occurrence frequency of various word types in a given position of text and r their rank order, that is, the order of their frequency of occurrence. A plot linking f and r yields a curve something similar to the hyperbolic curve in Figure 1.4. This is in fact a curve demonstrating Zipf's Law [Zipf 1949] which states that the product of the frequency of use of words and the rank order is approximately constant. Luhn used this hypothesis to enable him to specify two cut-

---

2 Free form text / Natural Language.

3 Tried and tested since the early days of [Luhn 1958].

offs, an upper and a lower bound (see Figure 1.4), thereby excluding non-significant words from an internal representation for a document. Words above the upper bound are considered to be common and words below the lower bound are rare and therefore not contributing significantly to the content of the document. Luhn thus devised a counting technique for finding significant words. Consistent with this he assumed that the resolving power[4] of significant words, reached a peak at a rank order bisecting the upper and lower cut-offs and from the peak fell off in either direction, reducing to near zero at the cut-off points. There are no hard and fast rules for determining where these cut-off points should be placed. They have to be established by trial and error.

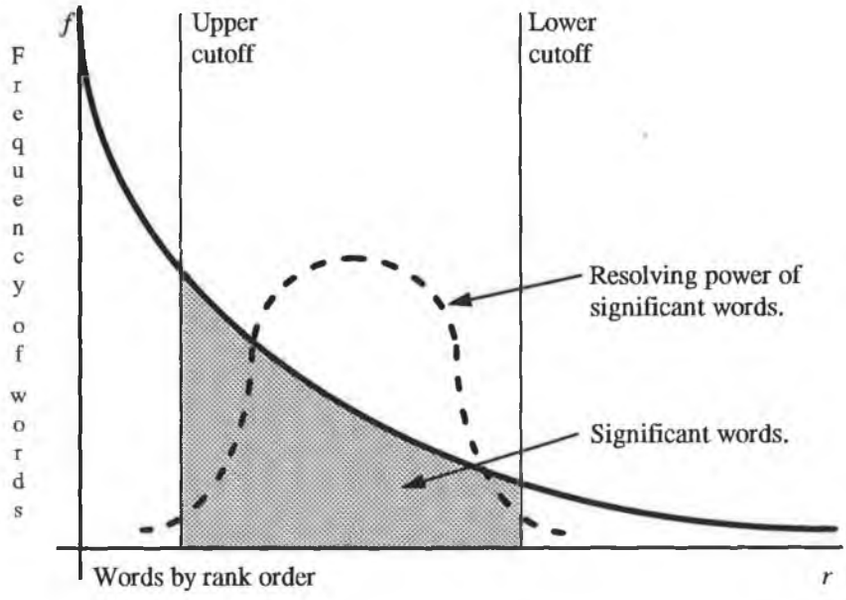


Figure 1.4 Hyperbolic curve relating occurrence frequency with rank order.

Luhn's ideas and assumptions form the basis for a significant portion of work to-date in IR. Luhn himself used these ideas in the process of generating automatic abstracts. There is no reason why the above principles should be restricted to only processing words, it could (and often has been) applied to word stems and to phrases.

[4] The ability of words to discriminate content.

1.7 Creating the Internal Representation.

The generation of the internal representation of the full text, abstracts or titles of documents by text processing systems ideally should be carried out using the minimum amount of human intervention. This is due to the vast amount of information that can potentially be processed by an IR system; any human interventions would slow the process down considerably. The internal representation of texts is simply the format in which it is depicted within the system. For example, an internal representation could be a simple list of class names, with each name representing a class of words occurring in the total input text. Such an indexing procedure will usually consist of three parts: 1). Removal of high frequency words. 2). Suffix Stripping. 3). Detection of equivalent stems.

The removal of high frequency words or *'stop words'* is one approach to implementing Luhn's upper bound cut-off. This can be achieved simply by passing the input text through a filter containing a *'stop list'* of words for removal. An extract of such a 'stop list' is illustrated in Figure 1.5. The advantages of using stop lists are twofold. Firstly, non-significant words are removed and therefore play no part in retrieval and, secondly, the size of the document being processed can be reduced by 30 to 50 percent.

A	Cannot	Into	Our	Thus
About	Co	Is	Ours	To
Above	Could	It	Ourselves	Together
see Appendix C for complete listing.				

Figure 1.5 Extract from a sample stop list.

The next stage in this process, suffix removal or conflation, is more complicated. A simple approach is to compile a complete list of all legitimate suffixes, match this list against the input stream of non-stopwords from the document and stem the non-stopword by removing the longest suffix. An example of such a list in illustrated in Figure 1.6.

-abilities	-alises	-ancial	-arisabilitv	-asisingful
-ability	-alising	-ancials	-arisable	-asisingly
-able	-alisingful	-ancies	-arisation	-asisings
see Appendix C for complete listing.				

Figure 1.6 Extract from suffix list.

Unfortunately this context free approach results in a significant error rate. For example, we may well want the '*ual*' removed from '*factual*' but not from '*equal*'. To avoid this problem a number of context rules must be defined in order to ensure that a suffix will be removed only if the context is right. For example:

- The length of the remaining stem exceeds a given number of characters; the default value is usually 2.
- The stem-ending satisfies a certain condition, e.g. does not end with the letter '*q*'.

Many words, which are equivalent in the above sense, map to one morphological form by removing their suffixes. Others, however, although they are equivalent, do not. For example, 'running' and 'ran'. It is the latter category which requires special treatment. The simplest approach to solving this problem is to compile a list of equivalent stem endings. In order for two stems to be equivalent they must match except for their endings, which themselves must appear in the list as equivalent. For example words stems such as '*absorb-*' and '*absorpt-*' are conflated because there is an entry in the list defining '*b*' and '*pt*' as equivalent stem endings if the preceding characters match. This is by no means a complete solution to the problem, it is in fact an over-simplification of the problem. For example words such as '*neutron*' and '*neutralise*' more than likely need to be distinguished from each other. There is no easy solution for this problem, it is one we put up with and assume that as a result system performance will not be adversely affected to any great extent. Perhaps the most well known implementation of an algorithmically based stemming procedure is [Porter 1980].

The final output from this process is a set of classes, one for each word stem detected. A class name is assigned to a document if and only if one of its members occurs as a significant word in the text of the document. An internal document representation therefore becomes a list of class names[5].

In summary, the raw data (text of documents) goes through a number of levels of processing in order to generate its internal representation. Initially we have the

document which is described as a string of words. The first step in the standardisation process is to remove the 'stopwords'. This results in a set of 'keywords' which are then passed through a conflation process the result of which is a set of classes or index terms. The next step in the process is the generation of index term classes by a process of automatic classification. In one sense this is where the normalisation stops. However, the use of index term weighting (See section 1.8) can also be considered as normalisation if the weighting scheme considers the number of different index terms per document.

It must be noted that the process used to generate the internal document representations is the same process used to convert the queries from their initial format to their internal representation with the retrieval process. This is necessary in order to achieve proper matches between the internal representations of the queries presented to the system and the documents indexed by the system.

1.8 Index Term Weighting.

We return to Luhn's idea of varying the discrimination power of index terms as a function of the rank order of their frequency of occurrence with the highest discrimination power being associated with the index terms with the highest occurrence frequencies. Luhn's use for this idea was the selection of significant terms from the text of a document. It is possible however to use his ideas to develop a weighting scheme for the individual index terms in a document. There is, in fact, a widely used weighting scheme which assigns each index term a weight directly proportional to its frequency of occurrence within the document. Initially, it may appear that this weighting scheme contradicts Luhn's ideas, however referring back to Figure 1.4, it would be consistent if the upper cut-off point is moved to the point where the peak occurs. It is highly probable that in fact this is what has occurred in experiments carried out using this form of weighting.

---

[5] Also referred to as index terms or keywords.

In addition to the above, attempts have been made to apply weighting based on the way index terms are distributed in the entire collection. The index term lexicon more often that not has a Zipfian distribution, i.e. if we plot the number of documents each index term occurs in according rank order we will get the usual hyperbolic shape. Work carried out by [Sparck Jones 1972] has shown experimentally that given a collection of N documents and an index term which occurs in n of them a weight of $\log(N/n)+1$ results in more effective retrieval than using no weighting at all. Assuming that indexing specificity is inversely proportional to the number of documents in which the index term occurs, the weighting scheme can be seen to be attaching more importance to more specific terms. The difference between these two weighting approaches can be summarised by stating that document frequency weighting emphasises the content description while specificity weighting emphasises the ability of terms to discriminate one document from another. Work by [Salton *et al* 1973] has yielded several conclusions Firstly, a term with a high total frequency of occurrence is not very useful for retrieval irrespective of its distribution. Secondly, mid-frequency terms are the most useful particularly if the distribution is skewed. Thirdly, rare terms with skewed distribution are likely to be useful but less so than mid-frequency terms. Fourthly, very rare terms are also quite useful but come bottom of the list except if they have a high total frequency. This introduces the notion of a 'term discrimination value' which measures the increase or decrease in the average dissimilarity between documents on the removal of that term. A 'good' term is one which, when used as an index term renders the documents within the collection more dissimilar. A 'bad' term has the opposite effect. The driving force behind these ideas is that a greater distance between documents will enhance the retrieval effectiveness.

1.9 Indexing Techniques.

Once the procedures for creating the internal document and query representations have been set in place we come to the next stage in the process, manipulating these internal document and query representations to achieve efficient and effective results. Efficient and effective in this context are the speed and quality of retrieval respectively. There are a number of existing and widely used approaches to manipulating these internal representations, for example:

- Full text scanning.
- Signature Files.
- Clustered Files.
- Inversion.

1.9.1 Full text scanning.

Full text scanning is the most straightforward way of locating documents containing specific search strings. A 'String' in this instance is a sequence of characters without 'Don't care characters'. If the query becomes complicated i.e. a boolean search expression involving many search strings, then an additional query resolution step is required to determine whether or not the term matches found by the substring tests satisfy the Boolean expression. Although simple to implement, this approach is far too slow to be practical in today's IR environment, for example, if x is the length of the search string and y is the length of the document (in bytes), then using a naive approach up to $f(x*y)$ comparisons are needed. Algorithms have been proposed [Knuth *et al* 1977] that need only $f(x+y)$ comparisons with a pre-processing time of $f(x)$. A fast string search algorithm was proposed by [Boyer & Moore 1977] where the idea is to perform character matches from left to right; if a mismatch occurs, the search string may be shifted up to x positions to the right. The number of comparisons is $n+m$ in the worst case and usually it is much less; for a random English pattern of length $x=5$, the algorithm typically inspects $z/4$ characters (where z is the starting position of the match). This string searching approach also required an $f(x)$ pre-processing time for the search string. In general, the main advantages of full text scanning approaches are that they incur no such overhead (no index required) and a minimal amount of effort is necessary for insertions and updates (no indices have to be changed). The price of these advantages is relatively poor response times especially for large text collections when compared to other indexing techniques. However full text scanning can play an important role in IR particularly in conjunction with other approaches such as inversion and signature files.

1.9.2 Signature files.

Interest has been expressed [Burkowski 1991] [Frakes & Bazea-Yates 1992] in using a signature file approach as an alternative to inversion for manipulating internal representations. In this method, each document yields a bit string or 'signature', through a process of hashing and then superimposed coding[6]. The resulting document signatures are stored sequentially in a separate signature file which is much smaller that the original text collection (typically between 10% and 20% the size of the information being indexed) and can be searched much faster. One problem with this approach is the fact that the signature file grows in linear proportion to the text collection. So for large text collections searching the signature file index eventually becomes a major overhead. Work has been done by [Lee & Leng 1989] and [Kelledy 1993] on methods for partitioning the signature file to reduce this problem. Other work [Lee 1987] has been carried out into modifying the signature file structure to attain efficiency improvements while maintaining the ease of update capability associated with this indexing scheme. In summary its advantages are a much smaller and easier to maintain indexing structure. Subsequent work carried out by [Kelledy 1993] has highlighted limitations with this approach with respect to retrieval efficiency. This coupled with the fact that limited memory and disk storage are not serious problems in today's IR systems when dealing with collection sizes up to the *TREC* collection size, favours the inversion approach as retrieval performance is better. The main strength of signature files is the simple file structure and ease of maintenance which is well suited to dynamic text collections and this, coupled with the fact that this approach is easily parallelised [Stanfill *et al* 1986], bodes well for this indexing scheme becoming popular with medium sized, dynamic text collections.

1.9.3 Clustered files.

The basic idea in clustering is that similar documents are grouped together to form clusters. The underlying reason for this is the so-called cluster hypothesis namely that 'Closely associated documents tend to be relevant to the same requests', which

can also accelerate searching by leaving less logical distance between related documents. Clustering has attracted much attention in the IR field [Salton *et al* 1983] [van Rijsbergen 1979]. It must be noted that clustering can be applied to terms as well as documents with terms grouped together and forming classes of co-occurring terms. These co-occurring terms are usually relevant to each other and are sometimes synonyms. Term grouping or clustering is useful in automatic thesaurus construction and in dimensionality reduction.

Document clustering involves two procedures, firstly, the cluster generation and secondly, the cluster search. A cluster generation procedure operates on vectors or points within a *t*-dimensional space (*t* being the number of permissible index terms) with documents represented by a vector which has index terms assigned to it during the indexing procedure. The values contained within the document vector are usually 0 if a particular term is absent or 1 (binary document vectors) or a positive number (term weight) which reflects the importance of the term for the document. The next step in the cluster generation procedure is to partition these document vectors into groups with the partitioning procedure ideally meeting two goals, these are that firstly, it should be theoretically sound and secondly, it should be efficient. The criteria for theoretical soundness are in essence as follows:

- The method should be stable under growth, i.e., the partitioning scheme should not change drastically with the insertion of new documents.
- Small errors in the description of the documents should result in small changes in the partitioning.
- The method should be independent of the initial ordering of the documents.

The main criterion for efficiency of the cluster generation process is the time required for clustering. Many cluster generation approaches have been proposed but unfortunately, no single approach meets both requirements for soundness and efficiency and this results in two classes of clustering approaches.

---

[6] Index term 'signatures' are overlaid on top of each other to form document 'signatures'.

- 'Sound' methods, that are based on the document-document similarity matrix.
- Iterative methods, that are more efficient and proceed directly from the document vectors.

Methods based on the similarity matrix usually require $f(y^2)$ time (or more) and apply graph theoretic techniques (y being the number of documents). A document-to-document similarity function which measures how closely two documents are related must also be defined. Given a document-document similarity matrix, a simplified version of such a clustering method would work as follows. First, an appropriate threshold is chosen and two documents with a similarity measure that exceeds the threshold are assumed to be connected by an edge. The connected components (or the maximal cliques) of the resulting graph are the proposed clusters. Retrieval can be further accelerated if we create hierarchies of clusters, by grouping clusters to form super-clusters and so on. One way to achieve this is by applying the above method for several decreasing values of the threshold.

Iterative methods operate in less than quadratic time, that is $f(y\ log(y))$ or $f(n^2/log(y))$ (y being the number of documents and n being the number of descriptors) on average. These methods are based solely on the document descriptions and do not require the similarity matrix to be computed in advance. The price of this increased efficiency is the loss of 'theoretical soundness'.

Searching clustered files is a much simpler process than cluster generation. The input query is represented as a t-dimensional vector and it is compared with the cluster-centroids which represent the central theme or focus of a document cluster. The searching proceeds from the most similar clusters, i.e., those whose similarity with the query vector exceeds a threshold. Structuring the collection in such a way will make the system more efficient (similar documents are physically close to each other and hence retrieval time will be quicker) and possibly more effective (any class found will tend to contain only relevant and no non-relevant documents).

1.9.4 Inversion.

Inverted files usually contain three main components. The first component is a dictionary file or lexicon which is simply a list of all index terms sorted in alphabetical order. Associated with each index term are a number of other important statistics, for example, the frequency count of the index term or in other words the number of unique documents it appears in within the document collection. Also a pointer or offset into a posting file must also be maintained. The second component of an inverted file structure is the postings file which contains lists of document identifiers, one list for each index term. There is also an option to include positional information i.e. the index term's position within the document.

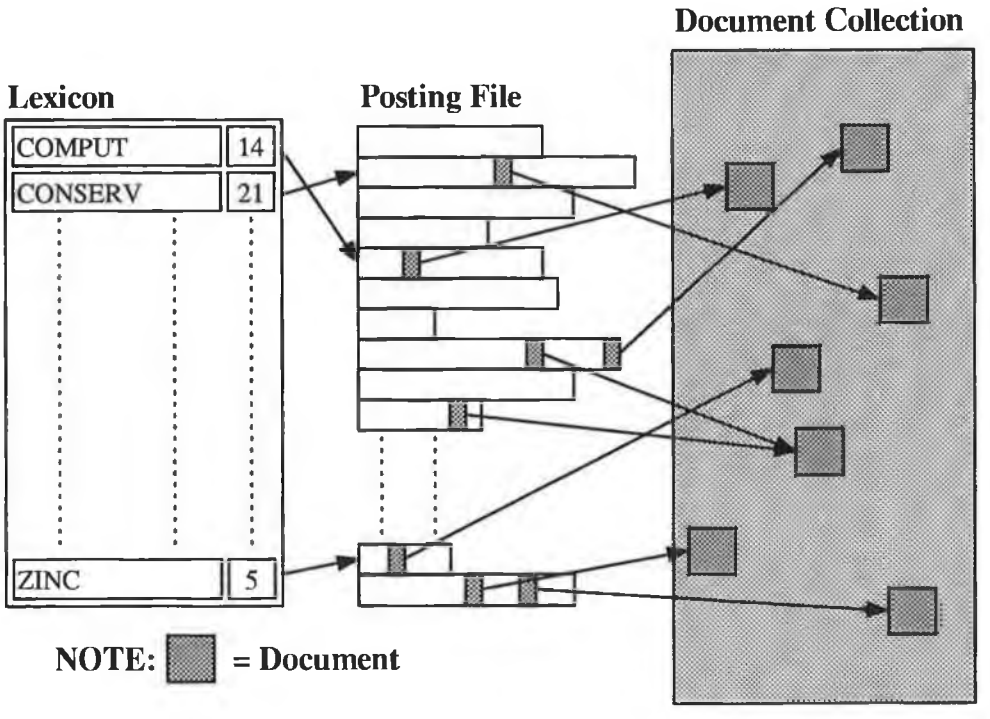


Figure 1.7 - General Structure of an Inverted File.

The third and final component of an inverted file structure is the raw information itself i.e. the documents which are being indexed. The vast majority of present day operational IR systems such as *DIALOG*, *BRS*, *MEDLARS*, *ORBIT* and *STAIRS* are based around inversion. More sophisticated methods can be employed to organise the lexicon, such as B-trees, TRIE hashing or variations and combinations of these. *STAIRS*, for example, uses two levels for the lexicon. Words starting with the same

pair of letters are stored together in the second level while the first level contains pointers to the second level, one pointer for each letter pair.

The first step in creating an inverted index is to take the internal representations of documents as described in Section 1.7 and use these to build the inverted file lexicon. Each entry in the lexicon points to its associated list in the posting file and each posting entry points in turn to a document.

The advantages of the inversion approach are numerous. Firstly, during the retrieval process, a minimum amount of information needs to be accessed in order to satisfy the query. Only documents known to contain query terms are accessed and used for further processing. This makes this approach the fastest on average of all tried and tested techniques. Secondly, quite sophisticated techniques can be incorporated into retrieval processes based on inversion i.e. additional information such as proximity and within document frequency and location information can be added into the posting file entries thus enabling very complex procedures, if required, to be added to the basic approach and thirdly, the inversion technique is relatively easy to implement.

The inversion approach also brings with it some disadvantages and one of these is that the storage overheads required to store such an index can be quite considerable. The index generally can occupy from 50% up to 300% of the size of the document collection being indexed. This figure can often tend towards the larger end of the scale if additional posting information is stored. Another disadvantage with inversion is that the structure of the index itself is quite complex resulting in maintenance (additions, deletions and modifications) being a non-trivial operation. The skewed nature of the distribution [Zipf 1949] of the postings lists results in a few index terms appearing very often, with the majority of index terms only occurring once or twice and this poses a challenge to the efficient processing of an inverted file. Techniques to minimise the effect of the above disadvantages of inversion [Faloutsos & Jagadish 1992] [Zobel *et al* 1992] have been developed.

Having outlined the above disadvantages it must be stated however that while it is still very important to maintain tight control over the index size and structure, today's disk storage problem is by no means as critical a problem as in years gone by,

thus relieving the pressure on finding indexing schemes which attain the performance levels of inversion without the storage overhead, even though seek times for such large index files is still a problem. The second disadvantage of structural complexity does not present itself as a major problem within our test environment (detailed in Chapter 4) which is essentially a static environment i.e. once the document collection is indexed no modifications to the index are required.

1.10 Problem being addressed in this thesis.

Today's demanding users require relevant information in response to their requests and need this information immediately. This coupled with the recent information explosion and society's increasing dependency on this information is motivating research into ways to meet these demands. To a certain extent computer hardware manufacturers are dealing with this problem via the development of advanced hardware based solutions, namely faster CPUs, larger amounts of main memory and disk storage available to the user and parallel architectures to name but a few. However as the old adage states *'A problem expands to fill the space and time allotted to it'*. This results in a race between technological developments on one side and increased demand and expectations from users on the other.

This problem is of acute importance to IR as more and more on-line information becomes available. In this sense IR is perhaps one of the most demanding computing disciplines with respect to storage required and speed of response to user information requests. Addressing this problem has been the subject of much research. [Persin 1994]. Such approaches address the efficiency and effectiveness issues concerning IR systems, efficiency being the speed of response to user information requests and effectiveness being the quality of that response. In most cases attempts at improving IR systems efficiency has resulted in a detrimental effect on the system's effectiveness.

It is our belief that there exist methods for attaining necessary levels of efficiency improvements without compromising the system's effectiveness. The body of research in this thesis will provide an in-depth analysis of the retrieval process, its underlying

structure and procedures plus the structure and nature of the *'Query Space'*[7] (QS) in an effort to highlight areas for algorithmic improvement and also identify regions within the QS of greater relative importance to the user's information requests. We envisage that advancing the solution to the above problem will in effect assist in redressing the imbalance in the race between technological developments and increased user demand and expectations. The exact details of our investigation into this area will be presented in subsequent Chapters.

1.11 Summary.

At this stage the reader should have a clear idea of what IR is and the context within which we are operating. Firstly, a brief history of the area followed by a detailed definition of the term *'Information Retrieval'* particular to our context was given. Secondly, a definition of what defines the area i.e. matching process used, inference type, type of query language, query specificity, to mention but a few is presented. This was followed by an overview of the sub-components that make up an IR system coupled with a functional description of each sub-component. Thirdly, sections dealing with generating and manipulating internal representations were discussed. Lastly, the problem being addressed was outlined and a statement of intent with respect to solving the above problem was presented.

Obviously the list of indexing techniques outlined in this Chapter is by no means an exhaustive one. Other tree and hashing based index structures exist. However in our opinion the above four approaches are the most suitable for the task in hand and less likely to 'fail' when handling the vast volumes of data required in today's IR environment.

The advantages and disadvantages associated with inversion as an indexing scheme coupled with the approach's flexibility and the fact that with little effort on our behalf this approach can be made to suit our needs perfectly, make it the logical choice as an indexing mechanism. In addition, the vast amounts of previous work using this

---

[7] Intermediate data generated during the retrieval process.

indexing approach from which we could draw from make inversion the ideal choice. A more detailed description of the exact inverted file structure and information contained therein will be presented in Chapter 4. Chapter 2 will present a detailed description of the major problems facing IR researchers in meeting users' ever more demanding expectations of IR systems.

2. Problem Definition.

2.1 Introduction.

The greatest challenge facing IR system developers and the systems they produce is one of their own creation, i.e. users of IR systems to date, be they basic string searching systems or more complex text indexing approaches, have seen the potential usefulness of such IR systems. For IR system developers the user interest in the systems they have developed has become something of a double edged sword in that once users have been exposed to such systems the next stage in the process is user feedback. This user feedback usually takes the form of *'Wouldn't it be nice if....'*, *'This aspect's good but...'*, *'I need more up-to-date information'*, in short users are becoming more discerning about what they want and expect from IR systems. The core thrust behind users' expectations is linked to today's rapidly moving environment in which information goes *'stale'* or out of date very quickly.

The IR systems which perform more 'intelligent' processing of user information needs take longer to complete. This coupled with the ever expanding amount of information being indexed (detailed in Section 2.2) is placing an ever increasing demand on IR systems. One solution to this situation is simply to regard it as somebody else's problem, namely, the computer hardware designers, i.e. wait until someone else has developed a machine that is fast enough and has enough memory to run the more sophisticated IR systems fast enough to meet users' expectations.

Computer hardware manufacturers are dealing partially with this problem through the development of faster CPU's, parallel architectures, larger amounts of memory and cheaper disk storage. However the ever growing amount of information being brought on-line is far outstripping the improvements in computer hardware. The fact is that the vast majority of technological developments are in response to user needs and not just developing the solution and then looking for the problem. This results in a competition between technological developments on one side and increased demand and expectations from users on the other.

As a result, it is our opinion that there is much room for research in the efficiency effectiveness trade-off of IR systems. It is critical that whatever IR systems do, they must do it in as efficient a manner as possible. The efficient handling of information is of acute importance to IR as more and more on-line information becomes available. In this sense IR is perhaps one of the most demanding computer related disciplines with respect to storage required and speed of response to user information requests.

2.2 The Information Explosion.

People have been complaining about the information explosion for years, but in some ways it is only just beginning. It was estimated in 1975 that some 50,000,000 books has been published up to that time. But the real problem is the rate of increase: it has been estimated that the amount of information in the world doubles every twenty months. The accuracy of these mind-boggling statistics may be debatable but they do serve to underline the problem that we all feel, the amount of on-line information is getting out of control.

The Internet is the world's largest computer network - a network of networks really - and one of its most popular and widely used services is the Usenet news service. This is a loose collection of news groups contributed to by a huge user community, and it's free. To give an idea of the information explosion on computer networks, Figure 2.1 shows how Usenet has grown in terms of both the daily number of news articles and the number of megabytes they represent. Even more alarming, though, is the rate of growth: the number of articles, newsgroups, megabytes, users, and computers on the Internet have all been increasing exponentially since statistics started being collected in late 1984. As Figure 2.1 shows, the Internet news traffic almost doubles each year. In fact the total Internet traffic is growing much faster; presently it rises by 12% each month, which corresponds to a doubling every six months. Clearly this cannot continue forever, there are some limiting factors. For

example, projecting the rate of growth of Internet users and the rate of growth of world population, the former will overtake the latter in the year 2000![8]

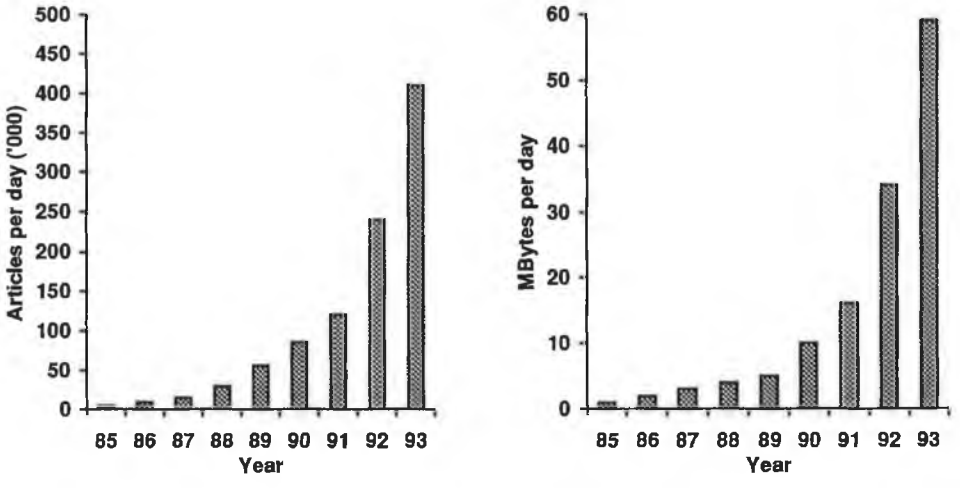


Figure 2.1 - Growth of News on the Internet.

Its hard to resist pointing to the Internet as the beginning of a phenomenon that might broadly resemble a world encyclopaedia. With 1.8 million computers (in mid-1993), each equipped with, say, 500 Mbytes of storage, it has been described as the worlds largest library. If just 5% of this disk space were allocated for network use, the total space would amount to nearly 50 terabytes (50,000,000 Mbytes). According to a 1993 estimate, the disk space occupied world-wide by the Internet news is half this amount (22 terabytes). Even this smaller figure is easily enough to accommodate a full-text database containing the text of the 50,000,000 books estimated to have been published by 1975, compressed and indexed.

2.2.1 Coping with the Information Explosion.

Finding information has always been difficult. Computer networks are certainly making it much easier, but along the way they are completely changing our expectations about what it is reasonable to try and find. For example, Internet users at the leading edge of technology now expect to be able to discover anyone's electronic

---

[8] To put this ridiculous projection into perspective, it is said that half of the world's population does not live within two hours' walk of a telephone.

mail address given the name and some vague additional clues ("somewhere in Europe"). They expect to be able to locate a file containing an interesting program given just the name of the program, or identify the latest papers on a particular specialised topic and immediately download them. Just a decade ago it would have seemed naive and unrealistic to predict that such incredible facilities for obtaining information would be in common use today.

There exists a large number of programs for structuring and locating information on the network. Archie is a system for locating publicly available files anywhere on the Internet. Gopher is a menu-based system for exploring Internet resources, and Veronica provides an index to the resources that Gopher makes available. The World-Wide-Web is a hypertext system for finding and accessing Internet resources. The proliferation of these programs testified to the extreme difficulty of finding what you want and the unreliability of the information present makes comprehensive retrieval mechanisms even more crucial, so that 'facts' can be not merely found, but checked and cross-checked as well.

Wide-area information service (WAIS) is a scheme that comes close to a full-text retrieval mechanism on the Internet. It can be thought of as a collection of private libraries that anyone can setup on the network and make available to others. Most are free, maintained by volunteers or public institutions, but some commercial information vendors provide their services through a WAIS interface, and for those a user must pay a fee to use it.

The idea of a 'Knowbot' or an 'Intelligent Agent' has emerged recently to assist with the task of finding information on the Internet [Maes 1994]. A Knowbot is an information retrieval tool, a robot librarian that 'knows' about different mechanisms for locating and retrieving information. Knowbots have been described as 'software worms' that crawl from source to source looking for answers to users questions. As they explore they may discover new sources, and these will be checked too. When a knowbot has exhausted all of its sources, it returns what it has found.

For example, the LifestyleFinder agent, newly released from Andersen Consulting's Agents research group, recommends URL's to users based on their

overall lifestyles. Another example is the WBI (Web Browser Intelligence) agent which acts as a WWW proxy between your browser and the rest of the Web. WBI can remember where you have been on the Web, what you found there, and can help you recall any word on any page you have visited. It can alert you before you go to a page, whether the site is not available or the access time will be slow and helps you navigate more productively through the Web by learning your preferences and patterns for searching for information. Yet another example is 'Smart NewsReader' from Intel. One of its features is that it can 'read through the articles' and score each thread of those articles based on a user's past interests. The articles can then be resorted based on this score. As the user reads articles he or she tells the system which articles they found interesting or boring.

2.3 Increased User Expectations.

In today's rapidly advancing information age the average user is becoming more and more computer literate. The 'fear' of computers is being eroded, especially in the younger generation. This has led to increased familiarity with and expectations from computer software in all its forms. It is no longer acceptable in today's competitive software industry just to develop a product that solves a user's need. The product must be seen to solve the problem in a stylish, easy to use and efficient manner. In the IR context, IR systems must effectively handle all of the user's idiosyncrasies such as vague and short information needs along with coping with the inherent ambiguous nature of the information being sought.

2.3.1 Vague User Information Needs.

In many situations in which IR systems are used users are not sure what they are looking for and they may need some help in formulating their information need. In this situation IR systems must direct users away from vague and imprecise terms towards specific and discriminating terms. Interactive IR systems are useful in this area where an initial vague query may be modified by a user in response to information returned by initial imprecise search. This query modification might be the elimination of some terms from the query due to them returning non-relevant information. This process is called relevance feedback and if incorporated properly into an IR system the relevance

feedback operation should incur very little extra overhead on the part of the user in formulating his or her subsequent information needs (modified in response to initial information returned). Thus relevance feedback can evoke a feeling of involvement on the users' part, i.e. the user feels that they are playing an important role in providing additional information. It must be noted that this not only applies if the users' extra involvement in the retrieval process yields positive results but also when the results are negative. As a user learns and becomes familiar with the interactive querying process he or she develops the ability to avoid mistakes that yielded negative results in the past.

2.3.2 Short User Queries.

Most users of IR systems don't want to spend a lot of time formulating their information need, the result of this being short queries limited to a few key words related (in the users opinion) to their information need. The overall performance in terms of effectiveness of IR systems participating in *TREC-3*[9] and *TREC-4* has illustrated that retrieval based on short queries (*TREC-4*) is not as effective as retrieval based on long queries (*TREC-3*). The view that the queries used up to and including *TREC-3* were too long and complex was one generally held by participants of *TREC-1*, *TREC-2* and *TREC-3*. As a result of this the queries used for *TREC-4* were much shorter in nature and were generally thought to be more a more realistic representation of a typical user information need. The average number of words (including stopwords) in a *TREC-2* query is 128.94 words, in *TREC-3* the average dropped to 105.28 words. The average dropped significantly in *TREC-4* to just 39.46 words per query. In TREC-5 participants have been given the choice of using short queries with an average of just 15.7 terms per query or longer queries with on average 80.88 terms per query.

---

[9] TREC (Text REtrieval Conference) is an annual benchmarking conference for IR systems funded by NIST (National Institute of Standards and Technology) and DARPA (Defence Advanced Research Projects Agency).

2.3.3 Ambiguity of Text and Information Needs.

Text by its very nature is ambiguous. Textual ambiguity can take the form of syntactic ambiguity, lexical ambiguity and semantic ambiguity [Smeaton 1995]. An example of syntactic ambiguity is the sentence "I saw her duck", did someone see her dive down to avoid a low-flying object, or did she show someone her feathered friend. Lexical ambiguity can be illustrated by the following two sentences "He *leaves* behind a great legacy" and "The *leaves* blew in the Autumn wind". The word '*leaves*' could be a form of the verb to leave, or the plural of the noun leaf. The following sentence "I noticed a man on the road wearing a hat" has two syntactic interpretations with the participial phrase "*wearing a hat*" modifying the man or the road. Semantic level interpretation should tell us that hats are worn by animate objects (men, women, etc.) and the latter interpretation (road) should be discarded.

It is easy to under-estimate the amount of ambiguity occurring in text due to the vast amount of experience and background knowledge we have accumulated during our lifetime. This experience and background knowledge gained through repeated everyday activities since early childhood provides us with a knowledge base from which we draw on to help us (with little or no apparent effort) disambiguate most texts.

The full complexity of the disambiguation process becomes apparent only when we try to automate this process. When humans are required to disambiguate a given term be it written or spoken we use the surrounding context to help us disambiguate the term. This context can be any number of things ranging from the surrounding language of the word being disambiguated to things like the setting in which the word was spoken, the tone of voice used to articulate the word, etc.

An automatic process simulating the human disambiguation process does not have any experiences or knowledge base on which to draw on to help it resolve the problem and hence finds it very difficult to effectively disambiguate ambiguous words. A possible solution to this problem is to construct a knowledge base in machine readable form from which a computer could extract the necessary information to aid effective disambiguation. A number of these machine readable knowledge bases have

been constructed they usually taken the form of a machine readable version of an already printed thesaurus. A notable exception to this is WordNet, a machine readable semantic knowledge base developed at Princeton University [Miller 1995]. Figure 2.2 details a sample output from the WordNet system, in this instance the noun 'Pen'.

Sense 1	pen -- *a writing implement with a point from which ink flows.* •=> writing implement -- *an implement that is used to write.*
Sense 2	pen -- *an enclosure for confining livestock.* •=> enclosure -- *a space that has been enclosed for some purpose.*
Sense 3	playpen, pen -- *a portable enclosure in which babies may be left to play.* •=> enclosure – *a space that has been enclosed for some purpose.*
Sense 4	penitentiary, pen -- *a correctional institution for those convicted of major crimes.* •=> correctional institution -- *a government-maintained detention facility.*
Sense 5	pen -- *female swan.* •=> swan – *stately heavy-bodied aquatic bird with very long neck and usu. white plumage as adult.*

Figure 2.2 - Senses of the Noun 'Pen'.

The senses are presented in the order of most frequently used first. Some senses are obvious (Sense 1), some are closely related to each other (Senses 2 & 3) and some are obscure (Sense 5). The above attempts at creating knowledge bases to aid computers in the disambiguation process are by no means optimal but they do represent a major step forward in this area. It has been shown [Richardson & Smeaton 95] that automatic Word Sense Disambiguation (WSD) currently operates at an 60% to 70% effectiveness level. To date most IR research incorporating automatic WSD has shown a net degradation in overall performance due to the incorrect word sense being selected by the system based on the context available. Other research [Sanderson 94] shows that in order for automatic WSD to be of benefit to IR systems in general it would need to be operating at an effectiveness level of over 90%.

The alternative to automatic WSD is manual WSD which is a very time consuming and subjective process. There exist a number of sample text collections in which every word had been manually sense disambiguated, for example the Brown

corpus (called SEMCOR in its disambiguated form). By all accounts this manual tagging was a laborious process and one which is infeasable for documents in today's IR environment with text collections of the order of 100's of Gigabytes. The only possible place at present for manual WSD is during query formulation. In this scenario the user would be asked to manually disambiguate any ambiguous terms they have entered in their query. This additional sense information supplied by the user could in turn be used to further refine the output of the IR system and provide more effective results as has been shown in an IR application for searching through textual image captions [Smeaton & Quigley 1996] and in an information filtering application, called BORGES [Smeaton 1996].

2.4 Handling User Expectations.

As already stated in Section 2.3, user's expectations of computer systems in general and in particular IR systems are increasing. In most situations users don't want to spend a great deal of time defining and formulating their information need in detail. As a result the vast majority of user's information needs are defined using less than 6 terms [Croft 1995]. The users still expect hi-quality results from such short queries even though the IR system has very little initial information to work with. In order to solve the problem of these two conflicting requirements, i.e. little initial information supplied and effective results required, some sort of internal 'magic' must be carried out in order to enrich the initial information need to a level at which effective results can be returned to the user.

Another major cause of failure in IR systems is vocabulary mismatch [Croft 1995]. What this means is that the same concept can be described using two totally separate vocabularies. For example, the sentence *"the kid struck the ground with a branch"* and the sentence *"the child hit the earth with a stick"* describe the same concept but have no key words in common. The same can occur in documents and users information needs in which the user is describing the concept in the document but not using the same terms to describe it.

These problems can be addressed using a process of automatic query expansion which is often regarded by the users as some form of 'magic' and as such is highly

desirable. This type of vocabulary expansion can result in the transformation of the document and information need representations, as with Latent Semantic Indexing, or it can be carried out using an automatic thesaurus built by corpus analysis.

There is also a danger in going too far in the process of simplifying the interface to IR systems. Users in general like to feel in control of the retrieval process. This feeling of control can be lost if the interface does not provide enough optional extras to help the user to manipulate the retrieval procedure. These optional extras should not interfere with the ability of the IR system to facilitate the quick and dirty entry of an information need by the user. A good example of this in operation is Digital's AltaVista search engine in which the user has the ability to quickly enter an information need and get results back almost instantly while there exist non-obtrusive options which allow the user spend more time defining more complicated and detailed information needs if required.

2.5 Impact on IR System Performance.

The implementation of some form of 'magical' internal processing in order to satisfy the conflicting user needs of ease of use and effectiveness of results inevitability has some impact on the efficiency of IR systems in terms of index time and query response time. Any automatic WSD or QE process requires some modifications to the internal representations of the documents and the information needs. These modifications effect the IR system in terms of the time taken to complete its task. IR system maintenance engineers would be concerned with the indexing time and the query response time. This index time is of major importance due to the nature of the text collections being indexed by IR systems today. Text collections which have a high document throughput i.e. a lot of additions and deletions have special requirements which must be met by an IR system's ability to efficiently handle these modifications. The critical measuring factor in indexing time is the number of Megabytes of text per hour the system can handle. The IR system maintenance engineer and the user in particular are concerned with the query response time. The addition of WSD and QE techniques into an IR system have an impact on the query response time. These techniques impose an extra load on IR systems both in terms of processing additional

automatically generated query terms and in more sophisticated processing on existing query terms.

In today's rapidly changing information environment the 'age' of the information returned to the user in also an important factor in its relevance. If the information delivered to the user in response to a request is out of date or 'old' information then its relevance is reduced even though the information may have been relevant to the user at some point in the past. It is critical in today's competitive business world where so must dependency is placed on having the most 'current' information possible that IR systems provide the right information at the right time.

It is not just enough to incorporate these additional sophisticated techniques into IR systems. These techniques must be implemented in such a way as to minimise their effect on retrieval effectiveness. It is our belief that other additional techniques should also be incorporated into IR systems which would offset against the additional processing costs incurred by WSD and QE without any degradation in retrieval effectiveness.

2.6 IR System Optimisation.

To address the problem stated above this body of research proposes to identify and develop optimisation techniques and methods which can be incorporated into an IR system and will increase an IR system's ability to employ more sophisticated effectiveness related techniques while at the same time improving the IR systems query response time. These optimisation techniques fall into the category of Query Space (QS) restriction and thresholding techniques which control and limit the amount of data processed and generated during retrieval.

2.7 Summary.

In this Chapter we outlined one of the major problems facing IR system today, namely, dealing with the conflicting user expectations of 'minimal effort on the users behalf during query formulation' and 'the right information returned at the right time'. We then described some existing techniques which are employed in order to compensate for the use of 'minimal effort on the users behalf' along with their

associated costs and impact on IR systems after which we outlined our proposed area of research in which methods that offset the costs of using sophisticated effectiveness related techniques are also incorporated into IR systems. In Chapter 3 we detail related research in the area of optimising IR efficiency.

3. Literature Review.

3.1 Introduction.

IR systems are constantly being challenged to manage larger and more complex document collections. Systems which have worked well to date will not necessarily continue to do so if they are not designed to cater for larger document collections and better retrieval performance on larger document collections requires more sophisticated retrieval techniques. Optimising efficiency during the retrieval process is of paramount importance because today, high quality results are not necessarily good enough unless they are delivered in an acceptable amount of time. If an IR system is too slow it may be intolerable to use, regardless of the quality of the results it produces.

Recent trends in the increase in volume and availability of information suggest that system speed will become more and more critical. There exists, at present, commercial document collections containing tens of Gbytes of information and this, coupled with digital libraries, may expand the size of these collections to the order of hundreds of Gbytes. As the size of document collections become larger and larger this poses two main problems for IR systems. Firstly, document retrieval becomes more expensive in terms of time taken and computing resources needed. Secondly, more sophisticated techniques are needed to identify relevant documents from these even larger collections. Unfortunately, more sophisticated retrieval techniques almost always imply more expensive retrieval thus further compounding the problems of providing high quality answers quickly and efficiently. Much research has been carried out into providing quality responses quickly, research which falls into two main categories;

1. Research at the indexing end of the problem i.e. developing new, more efficient, smaller and more flexible index structures. The main goal of this research direction is to provide improved approaches to indexing which reduce the overhead involved in maintaining indexes on such large document collections.

2. Research and the retrieval end of the problem, i.e. developing query processing techniques which handle the processing necessary to provide high quality responses

to information needs. This approach essentially involves addressing the efficiency / effectiveness trade-off which occurs when trying to provide high quality responses quickly.

3.2 Improving Index Flexibility and Reducing Index Overhead.

Improving the flexibility and extensibility of the indexing structure allows a more comprehensive internal document representation to be maintained within an IR system. The quality of the internal document representation has a direct impact on the quality of results obtained from the system and the preferred index type in nearly all IR search engines is the 'Inverted File' as it has been shown to be the most efficient in terms of retrieval performance. However the inverted file by its very nature is not the easiest of structures to handle efficiently. The root cause of this is its logical record length which is highly variable in nature. The second problem with the inverted file approach is the additional overhead required to hold the index, for example, an inverted file structure containing positional information (details about the position of the terms in the document) can be larger than the document collection being indexed. A great deal of research has been done on firstly, modifying the basic inverted file structure to one that is more amenable to update and secondly, reducing the overall size of the inverted file index itself.

3.2.1 Index Flexibility.

Next to the speed of retrieval the flexibility of the index structure is one of the most important criteria for measuring the performance of an IR search engine. Index flexibility covers all issues relating to the conversion of a document collection into an indexable form. The speed at which this is done along with the ability to handle dynamic i.e. rapidly changing document collections is becoming more and more important. To date most operational text collections tend to be more or less static in nature with the occasional periodic update. This type of collection can be easily handled by an off-line regeneration of an index on the updated collection and when this new index is created it simply replaces the old index. This form of index update can handle all the index update operations (add, amend and delete). Add and delete are achieved by simply adding or removing documents from the collection and the new

index when regenerated will automatically reflect such changes to the collection. The amend or modify operation is usually performed by an addition operation followed by a delete operation, i.e. the updated version of the document is added to the collection and the old document is then deleted. Again these changes will automatically be reflected in the newly generated index. This type of index update is the simplest approach to the problem of index maintenance however in today's IR environment it is usually infeasable due to the extremely large collection sizes being dealt with. What is needed is a way to update the index without the need to regenerate the entire index from scratch.

As mentioned earlier an inverted file index by its very nature is difficult to maintain due to its basic component, a variable length record (see Figure 1.7, Page 29). Document additions involve the insertion or appending of the new document's posting list entries in the correct position. Document deletions involve the identification of all posting list entries belonging to the document in question and flagging them for deletion at a later stage by a separate deletion and index compression or garbage collection process. Modifications are most easily accomplished by adding the internal representation of the updated document to the index and flagging the old version of the document for deletion from the index. The actual posting deletion and index compression process is one which need only be carried out periodically.

The above abstractly describes the process of updating an inverted index. However at a lower level there still exists a number of problems associated with 'growing' an inverted index efficiently. By 'growing' we mean the inclusion of the internal representation of new documents into the index which cause posting list lengths to increase. At a low level this can mean the insertion of data into the middle of a file and the shifting of existing data to make room, or the insertion of a internal file link to the location of the new posting data (usually at the end of the file). The inverted file could then be post-processed periodically to de-fragment its posting lists in order to improve I/O efficiency during the retrieval process.

Yet another approach to the index update problem is to batch documents to be updated and when a sufficient number of documents have been submitted for update, a 'delta' inverted file could then be created for all of the update documents. This 'delta'

inverted file would be searched during the retrieval process. Again a periodic process could be carried out to merge the existing index with the 'delta' index and the process would be repeated as required.

The flexibility of a basic inverted index structure is difficult to maintain. It becomes even more difficult when the index structure is enhanced or modified in some way in order to improve retrieval efficiency, for example, the use of compression techniques for reducing the overall size of the index. Updates into a compressed inverted file require the posting lists to be read into memory, decompressed, updated, re-compressed and written back out to disk.

It is felt however that modifications necessary to the indexing process in order to achieve improvements of efficiency and / or effectiveness at the retrieval end of an IR system are in most situations, worth the cost because it the is efficiency and / or effectiveness of the retrieval process that have the greatest influence on determining the user's overall opinion of the IR system.

3.2.2 Index Compression.

The index of a full-text retrieval system is one of the largest components of the system: when uncompressed, it may be 50%-300% of the size of the corpus being indexed [Linoff & Stanfill 1993]. The index itself may become particularly large if it includes positional information (e.g. section, paragraph, sentence and word) needed to support retrieval schemes which utilise proximity information. Compression is an attractive technology for reducing the size of the index. The key to compression is the observation that each inverted file entry is an ascending sequence of integers. For example, suppose that the term 'instrument' appears in eight documents within a collection, those numbered 3, 5, 20, 21, 23, 76, 77, 78. This term is then described in the inverted file by the entry:

< *instrument*;8;[3,5,20,21,23,76,77,78] >,

More generally, this stores the term t, the number of documents f_t in which the term occurs in, and then a list of f_t document numbers.

$$< t; f_t; [d_1, d_2, ..., d_{f_t}] >,$$

where $d_k < d_{k+1}$. Because the list of document numbers within each inverted file entry is in ascending order, and all processing is sequential from the beginning of the entry, the list can be stored as an initial position followed by a list of increments, the differences $d_{k+1} - d_k$. That is, the entry for the term above could just as easily be stored as:

$$< instrument; 8; [3,2,15,1,2,53,1,1] >,$$

No information has been lost, since the original document numbers can be obtained by calculating the sums of the gaps. The two forms are equivalent, but it is not obvious that any savings has been achieved. The largest gap in the second representation has the potential to be the same as the largest gap in the first, and so if there are N documents in the collection and a flat binary encoding is used to represent the gap sizes, both methods require $\lceil \log N \rceil$ bits per stored pointer. Consider each inverted file posting list as a sequence of gap sizes, the sum of which can be at most N. This allows improved representation, and it is possible to code inverted file posting list entries using on average substantially less than $\lceil \log N \rceil$ bits per entry.

Several specific models have been proposed for describing the probability distribution of gap sizes. These methods can be categorised into two broad areas, *global* methods, in which every inverted file posting list entry is compressed using the same common model, and *local* methods, where the compression model for each term's entry is adjusted according to some stored parameter, usually the frequency of the term. Local models tend to outperform global ones in terms of compression, and are no less efficient in terms of the processing time required during decoding, though they tend to be somewhat more complex to implement.

The drawback of compression is that fragments of the index must be decompressed at query time, which may have an adverse impact on response time and also the throughput of the IR system. Additionally in dynamic text collections the compression scheme must permit updates without excessive overheads. Much research

has been carried out by [Zobel *et al* 1992] and [Witten *et al* 1994] in the area of efficiently incorporating index compression techniques into IR systems.

3.3 Minimising Processing During Retrieval.

While a flexible and extensible index structure is a basic requirement for an efficient IR system it is by no means the only target in terms of achieving performance improvements in an IR system. Further improvements can be attained by efficiently processing whatever information is eventually included in the index structure. It is this area of research that has the most noticeable impact of the user's perception of an IR system's performance as in today's IR environment the retrieval process is almost always an online process with results needed and expected quickly. The indexing procedure is a background task usually restricted to a 'system administrator' type person. The two main criteria for measuring the efficiency of an IR system are its indexing speed (in Megabytes of text per hour) and its average response time to a user query (in seconds). Research in this area has concentrated firstly, on identifying and processing only the most important and discriminating sections of the query supplied by the user and, secondly, processing the minimum amount of information associated with each important and discriminating query term.

3.3.1 Query Term Restrictions.

Careful selection of the terms within the query that are actually processed during retrieval can have significant impact on both the effectiveness and efficiency of the retrieval process. Initially a query is usually a list of index terms each of which has an occurrence frequency associated with it which is particular to the collection being searched. This occurrence frequency has great bearing on the 'value' of the index term within the context of the current query. A more formal definition of the 'value' of an index term is its *Inverse Document Frequency* or *IDF* score which is as follows:

$$IDF = \log\left(N/n\right),$$

where N is the number of documents in the text collection and n is the number of documents the index term actually occurs in.

Terms that occur frequently within the text collection contribute relatively little in terms of discriminating power during retrieval, i.e. they do not help distinguish one document from another with respect to the query because they occur in so many documents. It so happens that these 'low value' terms also take up the majority of the processing time during retrieval because they occur so frequently. It therefore makes sense to try and eliminate these 'low value' index terms from the retrieval process as early as possible. Due to their large *IDF* weights, rarely occurring terms are likely to make large contributions to a document's final query-document similarity score and therefore will identify good candidate documents. More frequently occurring terms may still contribute significantly to the query-document similarity score of documents in which they appear frequently, i.e. they have large *term frequency* (*tf*) values. The partial query-document similarity score is usually a function or a variation of the basic *Term Frequency* by *Inverse Document Frequency* (*tf* * *IDF*). The core concept in query term restriction is to attempt to process only those terms with partial query-document similarity scores.

Some of the earliest optimisation work in IR was carried out by [Smeaton & van Rijsbergen 1981] in the context of the nearest neighbour retrieval model. An approach is described how an upper bound on the similarity of any unseen documents can be calculated based on the unprocessed query terms. If this upper bound is less than the similarity of the current best document then processing may stop. Work on improving the efficiency of the calculation of the nearest neighbour similarity was continued by [Murtagh 1982]. The output from this research was an approach that yielded results significantly better than anything reported on at that point in the IR area along with specific recommendations as to when this approach would be effective. Further work in this area is detailed in [Murtagh 1985] and [Murtagh 1993].

Work carried by [Brown 1995] and [Brown 1996] using the INQUERY system describes an approach for fast evaluation of 'structured' queries. An example of a 'structured' query in the context of this research is given in Figure 3.1.

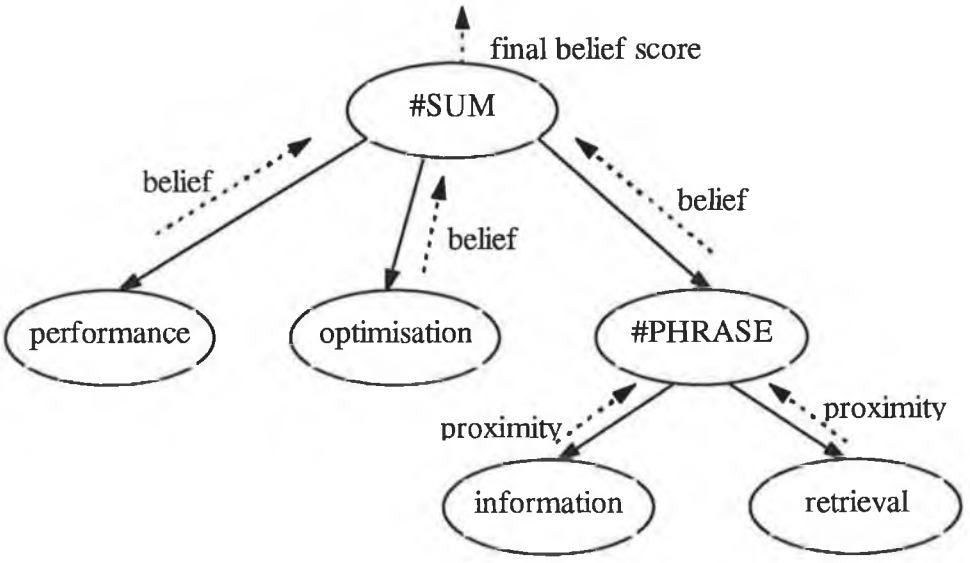


Figure 3.1 - Structured internal tree form of query in INQUERY.

Due to the structure of the INQUERY system in which a query is evaluated against the collection one document at a time rather than an index term at a time they use the structure (specific to the INQUERY system) to reduce the processing involved in responding to the query by reducing the number of documents that must be matched against the query.

Savings of up to 50% in execution time have been obtained with little drop off in effectiveness. The core idea here as with query term restriction is to terminate the processing of the query-document similarity score as early as possible if it is not likely to contribute to the overall result from the system. The common theme in the work carried out by [Smeaton & van Rijsbergen 1981], [Brown 1995] and [Brown 1996] is the reduction of the amount of information that needs to be processed in order to satisfactorily evaluate a users' query.

3.3.2 Posting List Restrictions.

The usual approach to the evaluation of ranked queries is consecutive processing of every term in a query and of the entire posting list for each of these terms. Using this approach a similarity between the document and query is determined for each query term and each document containing the term. This approach has a number of shortcomings. Usually a large percentage of the total number of accumulated partial

similarities are provided by commonly occurring terms and therefore have a low weight. Processing these low-scoring partial similarity values impacts very little on document rankings and consumes a significant percentage of retrieval time. This significant amount of retrieval time stems from the necessity in the vast majority of IR systems to decompress the indexing information before extracting the partial similarity scores and also the need to handle a far greater number of active document accumulators.

Dynamic stopping conditions have been proposed as a possible technique for improving the efficiency of an IR system [Persin 1994]. These approaches order the query terms by decreasing value to the query and process these terms until some stopping criterion is met. Work done by [Moffat & Zobel 1994] implemented the stopping condition by limiting the number of accumulators, firstly, by using a hard upper limit where by query processing was terminated when the upper limit was met and, secondly, by using a soft upper limit in which no new accumulators were added to the active set once the upper was reached.

This type of approach usually led to improvements in the retrieval time by significantly reducing the number of partial query-document similarity scores that were processed but also led to a corresponding deterioration in retrieval performance in terms of effectiveness. This can be attributed to the stopping condition being based only on global parameters of the document set.

Research carried out by [Persin 1994] implemented a more sensitive approach to processing the query term entries using the aforementioned global collection parameters and new local parameters (the number of occurrences of a query term in each document). To this end two thresholds were introduced into the retrieval process namely the accumulator insertion threshold t_{ins} and the accumulator addition threshold t_{add}. As each partial query document similarity score is computed it is compared first against the accumulator insertion threshold; if it is greater than the threshold then a new accumulator is added to the active accumulator set and is initialised to the value of the current partial query document similarity score. If the current partial query document similarity score is less than the accumulator insertion threshold then a check is made to see whether an accumulator already exists within the accumulator set for

the current document. If one exists then the current partial query document similarity score is compared against the accumulator addition threshold, if it is greater then the current partial query document similarity score is added to the existing value in that document's accumulator, otherwise the current partial query document similarity score is regarded as unlikely to contribute significantly to the result of the query being run and is discarded.

The novelty of this approach is that thresholds have been used to determine whether or not entire terms should be included or excluded from the retrieval process but not to determine whether individual query document similarity scores should be included or excluded. The values of both thresholds are determined as a product of a pre-defined constant and the accumulated partial similarity of the current most highly scored document. This heuristic approach supposes that if the current most highly scored document has a high weight then we do not need to process a document that has a small similarity value with query, as it is unlikely to change the final rank of scored documents or identify an important document that is not yet included in the set of scored documents.

$$t_{ins} = \frac{Const_{ins} * Sim_{max}}{f_{q,t} * \log(N/n)}, \qquad t_{add} = \frac{Const_{add} * Sim_{max}}{f_{q,t} * \log(N/n)},$$

In Persin's work $Const_{ins}$ and $Const_{add}$ which are used to define t_{ins} and t_{add} are pre-defined constants with $Const_{ins}$ always being greater than $Const_{add}$. Sim_{max} is the current maximum partial query document similarity score. The frequency of the term in the query is denoted by $f_{q,t}$ and $log(N/n)$ is the inverse document frequency of that query term with N being the total number of document in the collection and n being the number of these documents the query term occurs in.

This approach requires that the posting entries in the posting list are in order of decreasing within document frequency. This ensures that the most important entries in each posting list are processed first where important terms are those with a high within document frequency. It must be noted that while this sorting of the posting lists has certain advantages during the retrieval process by allowing some restriction of

processing the posting lists it also has implications for the index update procedure. The thresholds defined above are incorporated into the retrieval algorithm as follows:

1. Create an empty structure of accumulators.
2. Set the terms in the query in order of decreasing importance to the query.
3. Set the value of the weight of the current most highly scored document Sim_{max} to zero.
4. For each term t in the query.
   - 4.1. Compute the values of the threshold t_{ins} and t_{add}.
   - 4.2. If $t_{max} \leq \max(t_{ins}, t_{add})$, go to step 4.
   - 4.3. Retrieve the term entry for t from disk.
   - 4.4. For each (document d, term frequency $f_{d,t}$) pair in the term entry.
     - 4.4.1. If $f_{d,t} > f_{ins}$, add $w_{q,t} \cdot w_{d,t}$ to A_d and add A_d to the set of accumulators if necessary.
     - 4.4.2. Else, if $f_{d,t} > f_{add}$, add $w_{q,t} \cdot w_{d,t}$ to A_d if A_d is present in the set of accumulators.
     - 4.4.3. Otherwise go to step 4.
     - 4.4.4. Set $Sim_{max} = \max(Sim_{max}, A_d)$.
5. Divide each non-zero accumulator A_d by the document length W_d.
6. Identify the k highest values of accumulators and retrieve the corresponding documents.

The addition of a new document to the index is no longer a case of appending the new posting list entry to the end on an existing posting list. The new posting entry must be inserted into the posting list in the correct position with respect to its within-document frequency. Deletions and modifications to the index structure are also made more difficult by this sorting procedure because the location of a document's posting entries cannot be carried out via an efficient searching process since the posting lists are not keyed on any order of document identifier.

The re-sorted posting list allows the termination of posting list processing when a posting list entry's within document frequency is less than the addition threshold. No further posting list entries in that particular posting list need be processed as they are all guaranteed to be less than the addition threshold as well this principle is the same as that used in [Smeaton & van Rijsbergen 1981]. Using this approach additional information, namely the maximum within document frequency of a term in all of the documents must be stored in the index's lexicon. This small amount of additional information included in the lexicon allows further time saving during retrieval by

allowing a posting list to be accepted or rejected for further processing based only on the maximum within document frequency for that posting list. Thus there may be no need to access the posting list at all.

Persin's research was carried out using a sub-section of the *TREC* text collection and only articles from the Wall St. Journal were used (~517 Mbytes of text and 173,252 documents). His work showed the potential for reducing the amount of accumulators used during retrieval without significantly affecting the retrieval effectiveness of the system. He also evaluated the thresholding approach using different weighting schemes and the first weighting scheme evaluated was the cosine weighting scheme, defined as follows:

$$\cos ine_{q,d} = \frac{\sum_t sim_{d,q,t}}{\sqrt{\sum_t \omega_{q,t}^2} \cdot \sqrt{\sum_t \omega_{d,t}^2}}$$

where q is the query, d is the document, and $\omega_{x,t}$ is the weight of the term t in a document or query x. The expression $sim_{d,q,t}$ is the partial similarity between a query q and a document d given by the term t as follows:

$$sim_{d,q,t} = \omega_{q,t} \cdot \omega_{d,t}$$

The weight assigned to a term in a query or a document is determined using the frequency-modified inverse document frequency as described below:

$$\omega_{x,t} = f_{x,t} \cdot \log(N / f_t)$$

The second weighting scheme was one developed by [Lucarella 1988] which determined the similarity between a document and a query using the following formula:

$$S_{d,q} = \frac{\sum_t \omega_{q,t} \cdot \omega_{d,t}}{\sqrt{\sum_t \omega_{q,t}^2} \cdot \sqrt{\sum_t \omega_{d,t}^2}},$$

where q is the query, d is the document, and $\omega_{x,t}$ is the weight of the term t in a document or query x. The weight of a term is determined as

$$\omega_{x,t}=(0.5+0.5\cdot(f_{x,t}/f_{\max_x}))\cdot\log_2(N/f_t)$$

where $f_{x,t}$ is the number of occurrences of the term t in x, $f_{\max_x}$ is the maximum occurrence frequency among the terms associated with the document or query x, N is the number of documents in the collection and f_t is the number of documents containing t.

The third weighting scheme evaluated was a weighting scheme developed by [Harman & Candela 1990] which determined the similarity between a document and a query using the following formula:

$$S_{d,q}=\sum_t \frac{\log_2 f_{d,t}\cdot(\log_2(N/f_t)+1)}{\log_2 M_d}$$

where M_d is the total number of significant terms (including duplicates) in the document d. This similarity measure only considers the frequency of a term in documents and does not take into account the number of term occurrences in the query.

The primary focus of Persin's research was the development of a technique that allows fast evaluation of ranked queries while reducing the amount of main memory required during retrieval. This approach works best on databases used on small computers with acute limitations on the amount of main memory, CPU speed and disk access time. This however in our opinion is a questionable assumption in that handling medium to large text databases in modern computing environments would not be handled by 'small computers' with 'acute limitations' on main memory, CPU speed and disk access time in the first place.

3.4 Document Fragmentation.

In most IR systems developed to date the document is considered to be the most basic unit of retrieval, i.e. the IR system responds to the user's query with a list of documents (possibly ranked in order of probable relevance). This is not necessarily always the best approach and could be considered to be rather course grained. That is the unit of retrieval, the document, can be too large.

Provision of answers to informally phrased questions is a central part of information retrieval [Wilkinson 1994]. Answers traditionally come in the form of whole documents, but documents will often be unsatisfactory as answers. The document may be too large and unwieldy or the answer contained within the document is diffuse and hard to extract. It is also possible that word based retrieval may be misled by the breath of vocabulary in a long document. Consider for example the situation where two documents are relevant to a given query; document 1 is relatively short and its entire content is loosely related to the query, document 2 is much longer than document 1 however there is a passage or a section within document 2 that is highly relevant to the query. In this situation an IR system using the document as the basic unit of retrieval will probably rank document 1 above document 2 because a significant amount of text in document 2 in unrelated to the query. Even if document 2 is presented to the user the chances are that the user will read the start of the document and think (incorrectly) that it is non-relevant. This results in valuable information remaining undiscovered.

Much research as been carried out into solving this problem of variable document length and handling long documents. The obvious solution is to make the unit of retrieval (currently the document) smaller. This would mean that chapters, sections, paragraphs and possibly even sentences could become the basic unit of retrieval. This of course has implications for any IR system supporting such fine-grained retrieval. This most obvious result of reducing the retrieval unit size is that a much larger number of retrieval units are now required to cover the document collection. Consider a document collection with N documents, when the basic retrieval unit was the document we had the possibility of at most N units of retrieval being activated in response to a query. Once the retrieval unit's size is reduced then the IR system will be required to handle >> N units of retrieval even thought the volume of text may be the same. This has serious implications for the efficient operation of the IR system.

Another, more difficult problem to deal with is where to place the boundaries between the new units of retrieval. It is easy for the human reader to identify logical breaks in the flow of text by identifying the structure of the document (chapter

headings, section headings, paragraph boundaries and punctuation). The reader will also find it relatively easy to locate changes in context, even subtle ones, within the document by drawing on the vast wealth of general knowledge the reader has. The identification of changes in context is a difficult problem to solve automatically. One approach is to attempt to use the document structure itself to determine where context breaks occur. It is logical to assume that in most cases a context shift occurs when a new chapter or section heading is encountered. Paragraph boundaries can also be used to some extent for this purpose. In an ideal world this approach would result in good positioning of the logical breaks in the text flow however in reality there are as many different writing styles as there are authors. This fact causes complications for this simplistic approach.

The ideal solution would be an approach that could identify context shifts in free flowing text without needing to consider the document structure itself. It must be stressed at this stage that research into incorporating document paging or document fragmentation as it is also known has resulted in significant improvements in retrieval performance in terms of effectiveness and therefore is a line of research worth pursuing. There have been several proposed methods for forming document fragments. Obvious choices are sentences, paragraphs and pages. The minimum and maximum size of these documents fragments is also an issue for research. [Allan *et al* 1993] and [Salton *et al* 1993] have shown that the use of individual sentences can help determine the relevance of whole documents. The result of an inappropriate fragmentation strategy is the poor breaking of documents i.e. a break between document fragments occurring in the middle of a piece of text about a particular concept thus reducing the probability that it will be retrieved in response to a query about that concept.

In order to counteract this problem work carried out by [Callan 1994] using overlapping text fragments has been found to be useful. Innovative strategies tried by [Schäuble & Mittendorf 1994] have shown how hidden Markov models can be used to discover passages that can be used as retrieval fragments. Another alternative is to use the explicit *SGML* (Standard Generalised Markup Language) mark-up within the text itself. The motivation behind research in this area is the increasing lengths of documents in full-text collections. IR systems are being asked to handle larger and

more complex document structures in today's environment. This coupled with the fact that passage retrieval has been shown to improve effectiveness of IR systems makes this technique a core component of any modern IR system. The use of passage level evidence does raise questions such as how to define paragraphs and what is their role in long structured documents.

Matches at the document level are considered to be global evidence, while matches at the sentence level are considered to be local evidence. Matching at each level contributes is some way to the overall performance of the IR system. It therefore is logical to assume that some combination of evidence from different levels of a document's structure may provide better results than evidence from any single level.

The research carried out by [Callan 1994] was implemented on top of the INQUERY system which is a probabilistic information retrieval system. A number of approaches to the implementation of passage level evidence were tried out. These were discourse passages based on sentences, paragraphs and sections (derived from the document structure itself) and passages based on text windows (delimited by the number of index terms) of various sizes. The fragmentation approaches were tested on a number of test collections ranging in size from 3 Mbytes up to 2 Gbytes. Discourse and windowed passages are investigated in detail within his research

The really effective use of discourse passages requires more consistency from writers than other passage level evidence approaches, i.e. two writers describing the same subject would not only be likely to use different vocabulary to describe the subject but they would be likely to use completely different document structures. This inconsistency both in the vocabulary and the document structure used will lead to variations in the performance of discourse passages when incorporated into an IR system. Sloppy or rushed writing will result in paragraphs being inserted for padding only. Clearly it is impossible to force a consistency of writing style on authors. As a result discourse passages are likely to work well with highly edited encyclopaedia and newspaper texts but are likely to be unreliable when used with the like of news wire articles.

The last approach tried by [Callan 1994] was the use of windowed passages based simply on the window size in terms of index terms. This is the simplest approach to including passage level evidence in an IR system. This approach allows more control over the fragmentation process and will lead to more consistency in terms of passage size. With the first two approaches tried in this research there is no real upper limit on the size of the passage. While using the windowed approach an upper restriction is imposed limiting the maximum size of a passage.

The question of how to effectively combine evidence from different levels of evidence, in this case evidence at the document level and evidence at the passage level is looked into in Callan's research. The conversion to using passage level evidence is delayed as long as possible, i.e. the documents are still indexed as single units as before, it is only during the retrieval process that the conversion to passage level evidence takes place. This means that no modification to an indexing system is needed. Conversion to discourse passages was carried out by a set of heuristic rules (based on document indentation) by the system with conversion to windowed passages being carried out by the addition of an extra parameter in the query input. This on the fly conversion to passages by the system incurs an additional overhead during retrieval.

Two variations of the discourse passages were tried out by Callan, these were paragraph passages (based on the heuristic rules only) and bounded paragraph passages (based on the heuristic rules and combining numerous short paragraphs together). Results obtained by [Callan 1994] showed that paragraph passages (unexpectedly) performed poorly on collections of short and medium document length. The reason for this is due to the document structure of these short and medium length documents which contain paragraphs of one and two sentences without a corresponding shift in context, i.e. cosmetic structuring of the document rather than contextual structuring. Due to the poor performance of the paragraph passages the notion of grouping numerous short passages into bounded paragraph passages was introduced. Minimum and maximum paragraph sizes were introduced. The overall performance of bounded paragraph passages was better than the performance of real passages. Experiments using the windowed passages were also carried out. Due to the fact that the passages are computed on the fly the first windowed passage in a

document starts at the position of the first occurrence of a query term in the document with new passages of length n (n supplied with query) being created every n/2 words.

The splitting up of documents into constituent fragments is only half the passage level retrieval problem, the other half being how to best combine the query-passage similarity scores to achieve an accurate representation of the query-document similarity. Several approaches to combining query-passage similarities into query-documents similarities have been tried by [Salton *et al* 1993] with notable success. In [Callan 1994] however the combination of evidence was achieved via a weighted combination of the highest weighted individual passage and the weight of the document as a whole (document level evidence + 7 * passage level evidence). Due to the fact that performance improvements can be obtained by the combination of evidence at the document and paragraph levels there exists the possibility that combining numerous levels of evidence from sentences, paragraphs, sections, chapters right up to an entire document would result in even further performance improvements.

[Callan 1994] also carried out experiments to determine the optimal window size results show that a window size of 200 to 250 words provided consistently good results. One thing that must be considered is the possibility that passage level retrieval may be unsuitable for use with long queries as the chances of a passage matching a great number of terms in a small query are reduced, i.e. the use of long queries coupled with short passages would result in no passage being able to completely encapsulate a significant portion of a query. This would result in passages matching up against different aspects of the query. The problem would then be to determine which partial query / passage match is the most important.

On the implementation side of things the inclusion of passage level evidence leads to a 25% increase in execution time to rank passages as opposed to documents. As a result of work by [Callan 1994] and [Salton *et al* 1993] a number of interesting questions regarding what constitutes a 'good' passage have been raised. The overall performance of windowed, bounded, and unbounded discourse passages were evaluated with windowed passages performing the best as they are more independent of the writing style used in the creation of the documents.

The overall conclusion from research carried out in the area of passage level retrieval is that the use of passage level evidence results in slightly better performance than the use of document level evidence. However, the combination of evidence from different levels achieves significant improvements in effectiveness.

3.5 Relevance to this Research.

So far in this Chapter we have discussed the topics of index flexibility, indexing overhead, retrieval overhead and document fragmentation. Each of these topics coupled with today's rapidly evolving IR operational environment have a marked impact on what is needed for the design of an effective and efficient IR system. It is with these topics in mind along with research carried out to date in these areas that we have developed a number of overriding criteria which governed the development of our IR search engine, the most important of which is the efficiency of the IR system. As a result, our selection of what IR methods and techniques are included in our system are greatly influenced by the effort involved (in terms of real-time computing resources) to implement them.

The conclusions drawn by us from previous research are that IR system efficiency is greatly influenced by both index and retrieval overheads be they the amount of disk storage required for the index, the amount of temporary storage required during index creation, the number of disk accesses required during retrieval and the amount of memory required to hold temporary structures needed during retrieval. Index flexibility also has great bearing on the usefulness of an IR system in that high index flexibility coupled with low index overheads allows an IR system to keep up with changes in the information being indexed.

Document fragmentation has been proved to be beneficial in terms of improving IR system effectiveness by allowing more fine-grained retrieval. It does however impact on the IR system in that it increases the number of possible units of retrieval therefore increasing both indexing and retrieval overheads. This requires the careful design and incorporation of the document fragmentation approach into the IR system so as to minimise its impact in terms of efficiency.

The work carried out by [Persin 1994] in applying thresholds to the retrieval process initiated our research into minimising retrieval overheads both in terms of the amount of disk I/O and memory required. Persin's research highlighted the possibility of attaining significant improvements without significantly compromising effectiveness. Our research in the area of minimising overheads incurred during retrieval is dealt with in detail in Chapter 6.

The work carried out by [Callan 1994] and [Salton *et al* 1993] illustrated to us the fact that document fragmentation in today's demanding IR environment is becoming a necessity rather than an optional extra for effective retrieval. The approach to document fragmentation incorporated and used in our system was influenced greatly by its impact on system efficiency, with the approach selected having an easily quantifiable and controllable impact. The selection of our document fragmentation approach will be dealt with in detail in Chapter 5.

3.6 Summary.

In this Chapter we conducted an overview of certain aspects within the IR field as a whole in particular those aspects of IR which are of particular importance to this research, namely, index flexibility, index compression, query term restrictions, posting list restrictions and document paging. We then highlighted how the aspects of IR described in this Chapter and research carried out in these areas are of relevance in the context of our research. In the next Chapter we describe the experimental environment under which we tested our system.

4. Experimental Environment.

4.1 Introduction.

This Chapter will describe in detail the experimental environment in which our research was carried out. Firstly, a brief history of *TREC* along with *TREC*'s ideas and goals will be given. Secondly, a description of the text corpus we used along with a detailed description of the *TREC* document structure will be given. Thirdly, a description of the queries used in *TREC* experiments will be presented, followed by an outline of how the *TREC* relevance assessments were determined. Fourthly, some of the advantages and disadvantages associated with *TREC* will be outlined. We will then proceed to detail the specific subset of *TREC* used in our experiments.

4.2 Text REtrieval Conference (TREC).

In November 1992 the initial Text Retrieval Conference (*TREC*) was held at *NIST* [Harman 1994]. The conference, which was co-sponsored by *ARPA* and *NIST*, brought together information retrieval researchers to discuss how their systems performed on a new large test collection (the *TIPSTER* collection). This conference became the first in a series of ongoing annual conferences whose goal is to encourage research in retrieval from large-scale text collections and also to encourage increased interaction among research groups in industry and academia.

The research carried out by the participating groups in the five *TREC* conferences to date has been varied, but has followed a general pattern. *TREC-1*, in November 1992, required significant system rebuilding by most groups due to the huge increase in the size of the document collection. Up until then typical test collections such as CACM, NPL and INSPEC etc. were of the order of a few Mbytes in size. The *TIPSTER* collection occupies just over 2 Gbytes of space. By the time of *TREC-2*, August 1993, many of the original *TREC-1* groups were able to 'complete' their system rebuilding and tuning and as a result of this the *TREC-2* results show, in general, significant improvements over the *TREC-1* results. In some cases, however, the *TREC-2* results should be viewed as a baseline for more complex experimentation.

The *TREC-3* results in November 1994, reflect some of this more complex experimentation.

The first three *TREC* conferences were centred around two tasks based on traditional information retrieval modes, namely, a 'routing' task and an 'adhoc' task. In the routing task it is assumed that the same question is always being asked, but that new information is being searched. This task corresponds to that task performed by news clipping services or by library profiling systems. In the adhoc task it is assumed that new questions are being asked against a static set of data. This task is analogous to a researcher using a library, where the collection is known, but the information need of the researcher is unknown. Figure 4.1 outlines a typical *TREC* task.

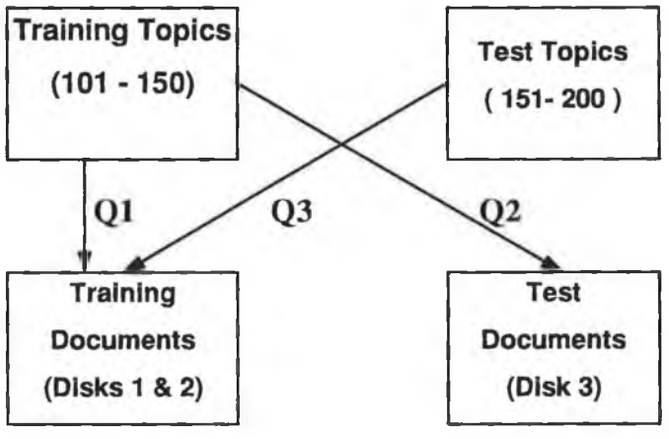


Figure 4.1 - A Typical *TREC* Task.

In *TREC* the routing task is represented by using known topics and known relevant documents for those topics, but new data for testing. This is illustrated on the left side of Figure 4.1. The routing participants are given a set of known (or training) topics shown in top left-hand box along with a set of known relevant documents (relevance judgements) for those topics. These topics are used to create a set of queries (the actual input to the system) which is then used against the training documents. This is represented by Q1 in the above illustration. Numerous sets of Q1 queries might be built to help adjust systems to this task, to create better weighting algorithms, and in general to train the system for testing. The results of this research are used to create Q2, the final routing queries to be used against the test documents, shown on the bottom right of Figure 4.1.

The adhoc task is represented by using known documents, but new topics with no known relevant documents. This is shown on the right-hand side of Figure 4.1, where the 50 new text topics are used to create Q3 as the adhoc queries for searching against the training documents. The results from searches Q2 and Q3 are the official test results sent to *NIST* by IR groups participating in the annual evaluation exercise.

In addition to clearly defining the tasks, other guidelines are used in *TREC*. These guidelines deal with the indexing and knowledge base construction methods and also the generation of queries from the supplied topics. Three generic categories of query construction were defined in *TREC*, based on the amount and kind of manual intervention involved.

- **Automatic:** Completely automatic query construction.
- **Manual:** Manual query construction
- **Interactive:** Use of interactive techniques to construct the queries.

The fifth *TREC* conference saw the addition of special tasks or 'tracks' each of which focused on a particular topic within the broad IR domain. These tasks were as follows:

- **Interactive**: investigating search as an interactive task by examining the process as well as the outcome.
- **Multilingual**: working with non-English test collections (250 Mbytes of Spanish text and 250 Mbytes of Chinese).
- **Natural Language Processing**: more focused investigation of NLP in an IR environment, emphasising the discovery and use of phrases for use in subsequent *TREC* experiments.
- **Multiple Database Merging**: investigation of techniques for merging results from various *TREC* sub-collections.
- **Data corruption**: examining the effects of corrupted data (such as would come from an OCR environment) by using corrupted versions of the *TREC* data.
- **Filtering**: evaluating routing systems on the basis of retrieving an unranked set of documents optimising a specific effectiveness measure.

TREC participants are able to choose from three levels of participation: Category A, full participation, Category B, full participation using a reduced dataset (1/4 of the full document set), and Category C, for evaluation only (to allow commercial systems to protect proprietary algorithms). All participants are provided with the data and asked to present one or two sets of results for each of the 50 topics. A set of results in this instance is defined as the top 1000 documents retrieved in response to a topic.

4.2.1 *TREC* Corpus.

Like most traditional retrieval collections, there are three distinct sections to the *TREC* collection, the documents, the questions or topics, and the relevance judgements or 'right answers'. *TREC* documents are distributed on CD-ROM's with about 1 Gbyte of data, compressed to fit, on each CD. The following table gives the document statistics of the *TREC* collection.

Subset of Collection	WSJ (Disks 1 & 2) SJMN (Disk 3) Ft (Disk 4)	AP	ZIFF	FR (Disks 1 & 2) PAT (Disk 3) FR94 (Disk 4)	DOE CR (Disk 4)
Collection Size (Mb)					
(Disk 1)	270	259	245	262	186
(Disk 2)	247	241	178	211	0
(Disk 3)	290	242	349	245	0
(Disk 4)	570	0	0	801	238
No. of Records					
(Disk 1)	98,732	84,678	75,180	25,960	2226,087
(Disk 2)	74,520	79,919	56,920	19,860	0
(Disk 3)	90,257	78,321	161,021	6,711	0
(Disk 4)	210,158	0	0	55,630	27,922
Median No. of Terms per Record					
(Disk 1)	182	353	181	313	82
(Disk 2)	218	346	167	315	0
(Disk 3)	279	358	119	2896	0
(Disk 4)	214	0	0	-	-
Average No. of Terms per Record					
(Disk 1)	329	375	412	1017	89
(Disk 2)	377	370	394	1073	0
(Disk 3)	337	379	263	3543	0
(Disk 4)	284	0	0	-	-

Figure 4.2 - Document Statistics for *TREC* (Disks 1-4).

Figure 4.2 illustrates some basic document statistics of the original *TREC* collection (Disks 1-3). For the *TREC-5* conference in November 1996 however new data was made available (Disk 4).

Although the collection sizes are roughly equivalent in megabytes, there is a range of document lengths across collections, from the very short (DOE) to the very

long (FR), the range of document lengths within a collection varies. For example, the documents from AP are similar in length (the median and average are very close), but the WSJ and ZIFF and especially FR documents have a much wider range of lengths within their respective collections. Figure 4.3 illustrates where the constituent parts of the *TREC* Collection are gathered from. The diversity of sources also adds value to the collection.

Disk 1	**Disk 2**
• WSJ: *Wall St. Journal* (1987, 1988,1989) • AP: *AP Newswire* (1989) • ZIFF: Articles from *Computer Select* disks (Ziff-Davis Publishing) • FR: *Federal Register* (1989) • DOE: Short abstracts from DOE publications.	• WSJ: *Wall St. Journal* (1990, 1991,1992) • AP: *AP Newswire* (1988) • ZIFF: Articles from *Computer Select* disks • FR: *Federal Register* (1988)
Disk 3	**Disk 4**
• SJMN: *San Jose Mercury News* (1991) • AP: *AP Newswire* (1990) • ZIFF: Articles from *Computer Select* disks • PAT: US Patents (1993)	• Financial Times • Federal Register (1994) • Computing Review Articles

Figure 4.3 - Document Sources for *TREC* Collection.

The documents are uniformly formatted into SGML. As can be seen in Figure 4.4 there are a number of tags which are common to all of the sub-collections making up the *TREC* corpus, these are the <DOC>...</DOC>, <DOCNO>...</DOCNO> and <TEXT>...</TEXT> tags. They denote the start and end of documents, the unique document identifier and the start and end of the text within a document. Each sub-collection has associated with it its own set of tags such as the <FTAG> in the Federal Register sub-collection.

Wall St. Journal

```
<DOC>
<DOCNO> WSJ870324-0001 </DOCNO>
<HL> John Blair Is Near Accord To Sell Unit, Sources Say</HL>
<DD> 03/24/87</DD>
<SO> WALL STREET JOURNAL (J)</SO>
<IN> REL TENDER OFFERS, MERGERS, ACQUISITIONS
(TNM) MARKETING, ADVERTISING (MKT)
TELECOMMUNICATIONS, BROADCASTING, TELEPHONE,
TELEGRAPH (TEL) </IN>
<DATELINE> NEW YORK </DATELINE>
<TEXT>
  John Blair &amp;amp; Co. is close to an agreement to sell its TV
station advertising representation operation and program
production unit to an investor group led by James H. Rosenfield,
a former CBS Inc. executive, industry sources said...
</TEXT>
</DOC>
```

Department of the Environment

```
<DOC>
<DOCNO> DOE1-01-0001 </DOCNO>
<TEXT>
The workshop was held to collect current data on the
experience with primary water stress corrosion cracking
(PWSCC) of steam generator tubing and the related laboratory
investigations. Thirty-two presentations were given covering
field experience, correlations of laboratory data on the field, and
relationship of material microstructure, stress, and environment
to PWSCC. The emphasis of the workshop was more on the
fundamentals associated with PWSCC yet culminated with
several presentations on remedial measures.
</TEXT>
</DOC>
```

AP Newswire

```
<DOC>
<DOCNO> AP890101-0001 </DOCNO>
<FILEID>AP-NR-01-01-89 2358EST</FILEID>
<FIRST>r a PM-APArts:60sMovies    01-01 1073</FIRST>
<SECOND>PM-AP Arts: 60s Movies,1100</SECOND>
<HEAD>You Don't Need a Weatherman To Know '60s Films
Are Here</HEAD>
<HEAD>Eds: Also in Monday AMs report.</HEAD>
<BYLINE>By HILLEL ITALIE</BYLINE>
<BYLINE>Associated Press Writer</BYLINE>
<DATELINE>NEW YORK (AP) </DATELINE>
<TEXT>
  The celluloid torch has been passed to a new generation:
filmmakers who grew up in the 1960s.
  ``Platoon,'' ``Running on Empty,'' ``1969'' and ``Mississippi
Burning'' are among the movies released in the past two years
from writers and directors who brought their own experiences of
that turbulent decade to the screen.
  ``The contemporaries of the '60s are some of the filmmakers
of the '80s. It's natural,'' said Robert Friedman, the senior vice
president of worldwide advertising and publicity at Warner
Bros...
</TEXT>
</DOC>
```

Federal Register

```
<DOC>
<DOCNO> FR89103-0001 </DOCNO>
<DOCID>fr.1-03-89.f2.A1000</DOCID>
<TEXT>
<FTAG tagnum=4700></FTAG>
<ITAG tagnum=90>
<T4>Federal Register</T4> / Vol. 54, No. 1 / Tuesday, January
3, 1989 /
Rules and Regulations
<ITAG tagnum=1>Vol. 54, No. 1</ITAG>
<ITAG tagnum=2>Tuesday, January 3, 1989</ITAG>
<ITAG tagnum=94>
<ITAG tagnum=69>
<ITAG tagnum=50>DEPARTMENT OF
AGRICULTURE</ITAG>
.....
This action is consistent with the marketing policy for 1988-89
adopted by the Navel Orange Administrative Committee
(Committee). The Committee met publicly on December 28,
1988, in Visalia, California, to consider the current and
prospective conditions of supply and demand and
recommended, by a ten to one vote, a quantity of navel oranges
deemed advisable to be handled during the specified week.....
</TEXT>
</DOC>
```

ZIFF Communications Company

```
<DOC>
<DOCNO>  ZF109-706-077  </DOCNO>
<DOCID>09 706 077.&O;</DOCID>
<JOURNAL>Business Week  Dec 31 1990 n3194 p93(12).&M;
</JOURNAL>
<TITLE>Fujitsu means business for America. (Special
Advertising Section by Fujitsu Ltd.) (Includes related articles on
the company's business relationships with Pepsi-Cola, Convex
Computer, Greenville EMS, and Sequent Computer
Systems)&M;
</TITLE>
<TEXT>
<ABSTRACT>In establishing itself as a major manufacturer in
the computer hardware market, Fujitsu Ltd boasts a long list of
corporate customers.&P;  The company's client base includes:
MCI Telecommunications Corp., Page Composition, Johns
Hopkins Hospital, Tiara Computer Systems Inc., Pepsi-Cola,
Convex Computer, Greenville EMS, and Sequent Computer
Systems Inc. The company stresses its good customer relations
and product development aspects, as well as its
telecommunications products.&O;
</ABSTRACT>
</TEXT>
<DESCRIPT>
Company:  Fujitsu Ltd. (Marketing).&O;
Topic:    Marketing Strategy
Customer Relations
photograph.&M;
</DESCRIPT>
</DOC>
```

San Jose Mercury News

```
<DOC>
<DOCNO> SJMN91-06364024 </DOCNO>
<ACCESS> 06364024 </ACCESS>
<CAPTION>  Photo; PHOTO: Associated Press; ANOTHER
TURNOVER -- Kansas City's Leonard
Griffin (98) closes in on Raiders quarterback Todd Marinovich,
who fumbled on
the play. Marinovich also threw four interceptions.
</CAPTION>
<DESCRIPT>  PROFESSIONAL; FOOTBALL; PLAYOFF;
GAME; RESULT; BRIEF  </DESCRIPT>
<LEADPARA>  Too much excitement on top of too much cold
medication may have caused the rapid heartbeat that forced
Kansas City linebacker Derrick Thomas out of the
...
reliable place-kicker, kicked an 18-yard field goal at 10:26 of the
fourth quarter, but he missed two field goals in the first half,
from 33 and 47 yards.
...
</TEXT>
<FEATURE>  PHOTO  </FEATURE>
<STATE>  CA  </STATE>
<WORD.CT>  539  </WORD.CT>
<DATELINE>  Sunday, December 29, 1991 00364024,SJ1
</DATELINE>
<COPYRGHT>  Copyright 1991, San Jose Mercury News
</COPYRGHT>
<LANGUAGE>  ENG  </LANGUAGE>
</DOC>
```

Figure 4.4 - Example Documents from various *TREC* sources.

4.2.2 *TREC* Topics.

In designing the *TREC* task, there was a conscious decision made to provide 'user need' statements rather than more traditional queries. Two major issues were involved in this decision, these are as follows:

- The desire to allow a wide range of query construction methods by keeping the topic (the need statement) distinct from the query (the actual text submitted to the system)
- The ability to increase the amount of information available about each topic, in particular to include with each topic a clear statement of what criteria make a document relevant.

Over the course of the *TREC* conferences to date a slight change in the above guidelines has occurred. The topics in *TREC-1* and *TREC-2* (topics 1-150) were not only very long, but contained complex structures. These topics were designed to mimic a real user's need, and were written by people who are actual users of a retrieval system. However they were intended to represent long-standing information needs for which a user might be willing to create elaborate topics, and therefore are more suited to the routing task than to the adhoc task, where users are likely to ask much shorter questions.

<top>
<head> *Tipster Topic Description*
<num> *Number: 101*
<dom> *Domain: Science and Technology*
<title> *Topic: Design of the "Star Wars" Anti-missile Defense System*
<desc> *Description:*
Document will provide information on the proposed configuration, components, and technology of the U.S.'s "star wars" anti-missile defense system.
<narr> *Narrative:*
proposed configuration, components, and technology of the U.S.'s "star wars" anti-missile defense system. The design and technology to be used in the anti-missile defense system advocated by the Reagan administration, the Strategic Defense Initiative (SDI), also known as "star wars." Changes of constituent technologies, are also relevant documents.
<con> *Concept(s):*
1. Strategic Defense Initiative, SDI, star wars, peace shield
2. kinetic energy weapon, kinetic kill, directed energy weapon, laser, particle beam, ERIS (exoatmospheric reentry-vehicle interceptor system), phased-array radar, microwave
3. anti-satellite (ASAT) weapon, spaced-based technology, strategic defense technologies
<fac> *Factor(s):*
<nat> *Nationality: U.S.*
</nat>
<def> *Definition(s):*
</top>

Figure 4.5 - *TREC*-2 Topic.

As a result of this the topics used in *TREC-3* (topics 151-200) are not only much shorter, but the complex structure of the earlier topics has been removed. In particular the <CONCEPTS> field was removed. This field contained a mini-knowledge base about a topic such as a real searcher might possess. The field was removed because it was felt that real ad hoc questions would not contain this field, and because inclusion of the field discouraged research into 'query expansion' i.e. techniques for expansion of 'too short' user need expressions. It must be noted that this change in topic structure poses no problem for the routing task, as experience in *TREC-1* and *TREC-2* has shown that the use of the training documents allows a shorter topic (or no topic at all).

In addition to being shorter, the *TREC-3* topics were written by the same group of users who performed the relevance assessments. Each of the *TREC-3* topics (151-200) were developed from a genuine need for information brought in by the assessors. Each assessor constructed his/her own topics from some initial statements of interest, and performed all the relevance assessments on these topics (with a few exceptions).

Figure 4.6 illustrates one of the topics used in *TREC*-3. Each topic is formatted in the same standard method to allow easier automatic construction of queries.

<top>
<num> *Number: 163*
<title> *Topic: Vietnam Veterans and Agent Orange*
<desc> *Description:*
While serving in South Vietnam, a number of U.S. Soldiers were reported as having been exposed to the defoliant Agent Orange. The issue is veterans entitlement, or the awarding of monetary compensation and/or medical assistance for physical damages caused by Agent Orange.
<narr> *Narrative:*
Relevant documents will discuss veterans suffering from cancer and other ailments allegedly caused by Agent Orange; the document will also relate the awarding of compensation to the veteran, or the veterans attempt to obtain compensation. Documents which discuss medically ailing children born to a veteran who had been exposed to Agent Orange are also relevant. Official studies are relevant, but articles which simply reference the Agent Orange problem are not relevant.
</top>

Figure 4.6 - *TREC-3* Topic.

After the *TREC-3* conference is was felt that the *TREC* topics were still too long and as a result the topics created for the *TREC-4* conference were shortened even further in order to accurately reflect the amount of effort a typical user is likely to

invest in the generation and specification of their information need as illustrated in Figure 4.7.

<top>
<num> Number: 201
<desc> Description:
What procedures should be implemented to insure that proper care is given to children placed under the au pairs' responsibility?
</top>

Figure 4.7 - *TREC-4* Topic.

<top>
<num> Number: 251
<title> Topic:
<desc> Description:
Documents will report the exportation of some part of U.S. Industry to another country.
<narr> Narrative:
Relevant documents will identify the type of industry being exported, the country to which it is exported; and as well will reveal the number of jobs lost as a result of that exportation.
</top>

Figure 4.8 - *TREC-5* Topic.

In *TREC-5* however participants were given the option of using short or long queries in their experiments with the short queries being a subset of the long query as illustrated in Figure 4.8.

4.2.3 *TREC* Relevance Assessments.

As is the case with all IR test collections relevance judgements are critical. For all topics under scrutiny it is necessary to compile a list of relevant documents and this list, hopefully, will be as comprehensive as possible. All *TREC* conferences to date have used the pooling method [Sparck Jones & van Rijsbergen 1975] to assemble relevance assessments. Using this method a pool of possible relevant documents is created by collecting a sample of documents selected by the various participating systems. This collection of possible relevant documents is then presented to the human assessors. More specifically, for *TREC*, the top 100 documents retrieved by each system for a given topic are taken and merged into a pool for assessment. This is a valid sampling technique since all the systems used ranked retrieval methods, with those documents most likely to be relevant returned first.

Evaluation of retrieval results using the assessments from this sampling method is based on the assumption that the vast majority of relevant documents have been found and that documents that have not been judged can be assumed to be non-relevant. A test of this assumption was carried out between *TREC-2* and *TREC-3* conferences, using the *TREC-2* results. Thirty six (18 adhoc and 18 routing) topics were selected for additional relevant assessments, using a pseudo random selection based only on the number of original relevant documents and on selecting equal numbers of topics from each assessor. For each selected topic, a new pool of documents was created by taking the top 200 documents from seven different runs known to achieve good results and to have little overlap in their document selection. New judgements were made on this pool, using the same judges who made the original decisions for each topic.

The following table illustrates the results of this experiment. On average, 30 new relevant documents (16%) were found for each of the topics, with a median of only 21 (11%) new relevant documents per topic. The median is much lower than the average because of the relatively large number of new documents found for those five topics with over 30% additional relevant documents found.

Percent New Rel	No. of Topics	Average New Rel.	Average Total Rel.	Average No. Jud.	Average 'Hardness'
0%	5	0	46	381	.3477
1-9%	11	10	173	257	.4190
10-19%	9	36	277	343	.2610
20-29%	6	47	185	190	.3660
40-33%	5	73	242	233	.5212
Average (over all 36 topics)		30	193	282	
Median		21	190	220	
Average (over 18 routing topics)		18	188	373	
Median		8	160	376	
Average (over 18 adhoc topics)		42	197	190	
Median		28	209	150	

Figure 4.9 - Analysis of Completeness of Relevance Judgements (*TREC-2*).

Figure 4.9 shows that there is some correlation between the number of new relevant documents found and the original number of relevant documents, particularly in that topics with few relevant documents to begin with tended to have few new ones found.

	Adhoc			Routing		
	Possible	**Actual**	**Relevant**	**Possible**	**Actual**	**Relevant**
TREC-1	3300	1279 (39%)	277 (22%)	2200	1067 (49%)	371 (35%)
TREC-2	4000	1106 (28%)	210 (19%)	4000	1466 (37%)	210 (14%)
TREC-3						
at 100	4800	1005 (21%)	146 (15%)	4900	703 (14%)	146 (21%)
at 200	9600	1946 (20%)	196 (10%)	9800	1333 (14%)	187 (14%)

Figure 4.10 - Overlap of Submitted Results.

In contrast, there is no correlation between the number of new relevant documents and the number of new judgements made, or between the number of new relevant documents found for a topic and the 'hardness' of the topic (a measure of the average system performance for that topic).

An alternative measure of the effect of pooling can be seen by examining the overlap of retrieved documents found from the various IR system participating in *TREC*. Figure 4.10 shows the statistics of the merging operations in the *TREC* conferences to date.

***TREC-2*: Relevant Documents Found in 'Second' Run**			
Percent New Rel.	**No. of Topics**	**Average New Rel.**	**Average No. Rel**
0%	0	-	-
1-9%	6	9	123
10-19%	19	26	163
20-29%	19	68	274
30-36%	5	109	296
Average		48	210
Median		30	201

***TREC-3*: Relevant Documents Found above 100**			
Percent New Rel.	**No. of Topics**	**Average New Rel.**	**Average No. Rel**
0%	1	0	85
1-9%	12	3	65
10-19%	7	13	96
20-29%	22	59	237
30-36%	8	137	381
Average		50	196
Median		30	122

Figure 4.11 - Pooling Analysis (adhoc).

For *TREC-1* and *TREC-2* the top 100 documents from each run (33 runs in *TREC-1* and 40 runs *TREC-2*) could have produced a total of 3300 and 4000 documents to be judged (for the adhoc task). The number of unique documents

(actually judged) was 1279 (39%) for *TREC-1* and 1106 (28%) for *TREC-2*. It must be noted that even though the number of runs increased by 20% (adhoc), the number of unique documents found has actually dropped but the percentage of relevant documents has not changed much. The more accurate results moving from *TREC-1* to *TREC-2* mean that fewer 'noisy' nonrelevant documents are being found by the systems. This trend continued into *TREC-3* even though the pooling method was changed.

***TREC-2*: Relevant Documents Found in 'Second' Run**			
Percent New Rel.	**No. of Topics**	**Average New Rel.**	**Average No. Rel**
0%	40	0	6
1-9%	8	4	61
10-19%	21	33	220
20-29%	11	88	345
30-36%	6	84	259
Average		44	210
Median		33	163

***TREC-3*: Relevant Documents Found above 100**			
Percent New Rel.	**No. of Topics**	**Average New Rel.**	**Average No. Rel**
0%	7	0	24
1-9%	9	6	106
10-19%	16	19	129
20-29%	16	94	354
30-36%	2	91	249
Average		41	187
Median		13	123

Figure 4.12 - Pooling Analysis (routing).

In *TREC-3* due to expected constraints on relevance accessor time, only one run from each participant was judged (participants specified which of their submitted runs they wished assessed). What happened was due to increased overlap between submitted results (outlined above) and more efficient judging by the relevance assessors extra time became available. As a result of this, the decision was made to judge the top 200 documents from submitted runs as opposed to the top 100 only. Figure 4.10 presents the results of the *TREC-3* mergings at both 100 and 200 documents. The percentage of unique documents found continues to drop compared with *TREC-2*. The drop in the total number of relevant documents over the conferences to date has dropped marginally. This is due to a deliberate tightening of the topics between *TREC-1* and *TREC-2*. Figure 4.10 also illustrates the drop in relevant documents found beyond the 100 document boundary. This not only reflects

the ranking performed by the systems, but also shows the diminishing numbers of relevant documents to be found even as the judged pool continues to grow.

The use of a different pooling method in *TREC-3* provided a chance to compare the two methods. Figure 4.11 and Figure 4.12 illustrate this comparison. The first method (used in *TREC-2*) took the top 100 documents from the two runs, whereas the second method (used in *TREC-3*) took the top 200 documents from a single run. The common ground for both methods in the top 100 is the single or 'first' run. The additional documents to be compared are the number of relevant documents in the top 100 for the 'second' run (*TREC-2*) versus the number of relevant documents in the second 100 in the single run for *TREC-3*.

Figure 4.11 illustrates that both pooling methods worked equally well for the adhoc task. For the routing task however, Figure 4.12 illustrates that the first pooling method (*TREC-2*) seems to have found more relevant documents (higher median). This could reflect the change in topic structure between *TREC-2* and *TREC-3*, however it is more likely a reflection of the difference between system performance in the adhoc and routing tasks. The above analysis suggested a return to the *TREC-2* pooling methodology was in order for *TREC-4*. Participating groups also preferred relevance judgements on both official runs as this allows more precision in evaluating run variations.

4.2.4 Advantages / Disadvantages of TREC.

As stated in Section 4.2, the *TREC* text collection was set up in response to the need for a large scale text database complete with queries and most importantly corresponding relevance judgements. Up until the *TREC* collection was created, most IR research was carried out using small and unrealistic test collections, such as the CACM, NPL and INSPEC collections which are in the order of a few Mbytes in size. The limitations of these collections were obvious in that they did not facilitate research into how IR systems would perform in operational environments in which the volume of information they would be expected to deal with would be many orders of magnitude bigger than the test collections previously encountered. By its creation *TREC* changed all this by providing a realistic test collection for IR researchers to get

to grips with. The most important part of the *TREC* collection is not the collection itself but the collection's relevance judgements. These relevance judgements allow researchers to tune their systems in order to achieve optimal performance in test environments before running their systems live in operational environments.

Like any other ground breaking endeavour *TREC* has not been without its drawbacks. At its creation five years ago the collection size of just over 2 Gbytes seemed very adequate for testing purposes. However today 2 Gbytes is starting to look small when compared against the information explosion we are witnessing on the World Wide Web. This problem however is currently being tackled by the *TREC* organisers by the inclusion of an additional 'track' for the next *TREC* conference. This track, is provisionally titled 'The Very Large Collection' track and is hoped to be based on a collection of 20 to 30 Gbytes of text. The idea behind this track is to see how existing IR systems will scale up to such volumes of information and to point out any unforeseen problems that are not obvious when dealing with the current collection size.

Another drawback of the *TREC* collection in the early years (*TREC-1* to *TREC-3*) was the format of the topics. Many *TREC* participants felt they were too long and detailed. It was felt that the topics made it too easy on IR systems to get good results and that IR systems in an operational environment would be unlikely to have to deal with such finely described information needs from typical users. As the *TREC* conferences progressed the *TREC* topics have become shorter and shorter in order to more accurately reflect typical user information needs. These shorter information needs provided more of a challenge to the IR systems carrying out the *TREC* task(s).

4.3 Evaluation of Results.

Much effort has been invested in addressing the problem of evaluating IR systems [van Rijsbergen 1979]. The question of what must be evaluated in order to constitute an effective evaluation of the operation of an IR system has been answered as early as 1966 by Cleverdon. He listed six measurable quantities, these are as follows:

- The *coverage* of the collection, that is, the extent to which the system includes relevant matter,
- The *time lag*, that is, the average interval between the time the search request is made and the time the answer is given.
- The form of *presentation* of the output.
- The *effort* involved on the part of users in obtaining the answers to their search requests.
- The *recall* of the system, that is, the proportion of relevant material actually retrieved in answer to a search request.
- The *precision* of the system, that is, the proportion of retrieved material that is actually relevant.

The last two evaluation criteria (recall and precision) bring in the notion of 'relevance' which is in itself a subjective notion, i.e. different users may (and probably will) differ about relevance or non-relevance of particular documents to given questions. The subjective quality of relevance can usually be circumvented by using bona fide users (users in a particular discipline with an information need) and having the relevance assessments made by a panel of experts in that discipline. This results in a situation (see *TREC*) where a number of questions exist for which the 'correct' responses are known. It is a generally held view in the IR field that IR systems that fare well under a large number of experimental conditions are likely to fare well in operational situations where relevance in not known in advance.

The effectiveness of an IR system is its ability to satisfy the user in terms of the relevance of documents retrieved and is traditionally measured by precision and recall.

	Relevant	Not Relevant	
Retrieved	$A \cap B$	$\overline{A} \cap B$	B
Not Retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$	$\overline{B}$
	A	$\overline{A}$	N

(N = Number of documents in the system)

The table above details all possible states which can occur after a retrieval operation with respect to documents and relevance. $A \cap B$ is the set of relevant

documents that are retrieved by the IR system. $\overline{A} \cap \overline{B}$ is the set of non-relevant documents that are not retrieved by the system. Using this table a number of effectiveness measures can be defined, as follows:

$$PRECISION = \frac{|A \cap B|}{|B|},$$

$$RECALL = \frac{|A \cap B|}{|A|},$$

$$FALLOUT = \frac{|\overline{A} \cap B|}{|\overline{A}|},$$

There is a functional relationship between the above effectiveness measures involving a parameter called generality (G) which is a measure of the density of relevant documents in the collection. The relationship is as follows:

$$P = \frac{R \times G}{(R \times G) + F(1-G)}, \text{ where } G = \frac{|A|}{N}$$

For each request submitted to an IR system one of these tables can be constructed after each returned document has been evaluated for relevance. Based on each one of these tables a precision-recall value pair can be calculated. If the output of the retrieval strategy depends on a parameter, such as rank position (position in top 1,000 documents for *TREC*), it can be varied to give a different table for each value of the parameter and hence a different precision-recall value. If λ is the parameter, then P_λ denotes precision, R_λ denotes recall and a precision-recall value pair will be denoted by the ordered pair (R_λ, P_λ). The set of ordered pairs makes up the precision-recall graph (see Figure 4.13). Geometrically when the points have been joined up in some way they make up the precision-recall curve. The performance of each request is usually given by a precision-recall curve. To measure the overall performance of an IR system, the set of curves, one for each request in the test collection of queries, is combined in some way to produce an average curve.

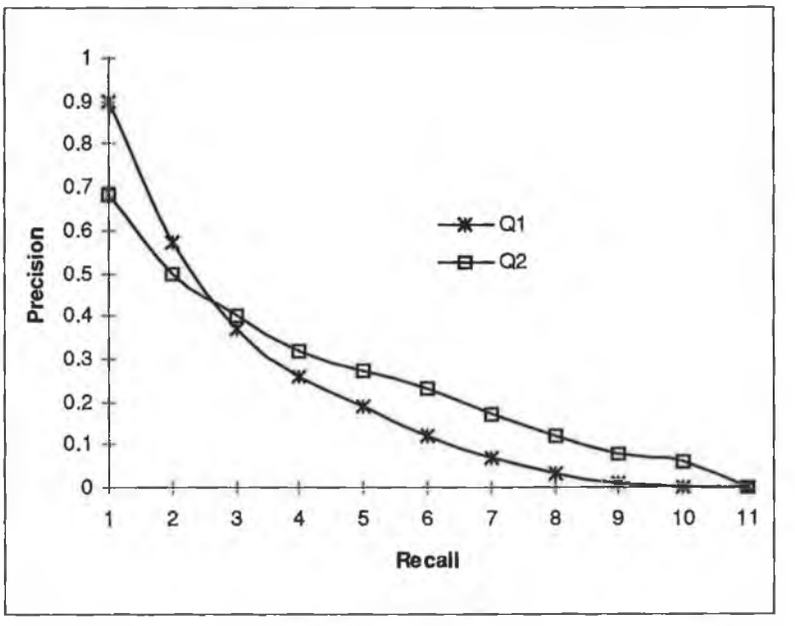


Figure 4.13 - Sample Precision-Recall Curves for two Queries.

Cleverdon's first four measurable quantities with respect to IR system evaluation, namely, *coverage*, *time lag*, *presentation* and *effort* are also important criteria when evaluating the performance of an IR system. Of these four criteria the time lag is the most straightforward and the easiest to quantify. It is simply how quickly the system responds to user requests. Evaluation of this criteria forms a significant part of the experiments carried out during the course of our research because we believe that IR system response time plays a critical factor is determining a user's overall opinion on the effectiveness of an IR system.

The *effort* criterion also played a significant part in the development of our IR prototype. We believe that in general most users are a) not willing and b) not sufficiently trained to invest a great deal of effort into the formulation of their information needs. (This was a perceived flaw in the early *TREC* query sets, they were too detailed. It was felt that users would never invest so much effort formulating such long and detailed information needs). As a result of this we geared our system towards minimising the amount of user effort required to formulate and run a query.

While we recognise that the form in which the results of an IR system are presented to the user is also very important to the overall user perception of an IR system, we decided to focus our efforts on the other evaluation criteria and to pay less

attention to the *presentation* of the results of our system. At present the system produces a ranked list of document identifiers to the user in response to their information need. This was sufficient for our experimental needs within the *TREC* test environment.

The issue of *coverage* was also not critical to our research as the *TREC* collection with its associated query sets and corresponding relevance judgement sets deals with the *coverage* issue in a rigorous and effective manner.

4.4 Summary.

In this Chapter we detailed the experimental environment in which our research has been carried out. We described the reasons for the creation of the *TREC* collection and its associated topics and relevance judgements. We outlined the advantages and opportunities that the *TREC* collection offers IR researchers as it is currently the most realistic IR test environment available. The statistics of each of the sub-collections within the *TREC* corpus were presented along with our reasons for using these sub-collections. How IR systems can be evaluated was described along with the effectiveness evaluation approach adopted by *TREC* (Precision-Recall Graphs). The next Chapter describes the IR system developed during the course of our research.

5. System Description.

5.1 Introduction.

In this thesis we will report on the research we carried out in the area of efficiency optimisation in IR systems and also the results we obtained from our research. However in order to validate our research and results we needed to implement and evaluate our IR efficiency optimisation techniques. This was done by developing an IR system during the course or our research called 'InfoLore'. This Chapter describes in detail the two main components of that system. Our IR system can be separated into two main sub-systems, namely the indexing sub-system and the retrieval sub-system. These sub-systems have rather diverse characteristics yet both are required to operate efficiently in order to provide the IR system user with effective and efficient responses. Section 5.2 outlines the operation of all of the components of the indexing sub-system.

- Statistics gathering.
- Document pre-processing.
- Partial index creation.
- Partial index merging.
- Index post-processing.

Section 5.3 outlines the operation of all of the components of the retrieval sub-system which are as follows:

- Query pre-processing.
- Pre-compute phase.
- Inverted file access.
- Normalisation and ranking.
- Output of the results.

5.2 The Indexing sub-system.

In today's IR environment where collection sizes are increasing exponentially and the text collections themselves are highly dynamic (subject to very frequent additions, deletions and modifications), the efficiency and effectiveness of an IR system is greatly effected by the time taken firstly to create an index and secondly to update it. It is therefore essential when designing an IR system to cater for the ability to 'grow' a text collection's index in an efficient manner in order that changes to the text collection itself are reflected in the collection's index as soon as possible. It is with these criteria in mind that we designed our indexing system.

A number of processing stages must be gone through in order to transform the text collection in its original form (free text) into a structure which facilitates the effective and efficient indexing of that text collection (in our case an Inverted File). These processing stages are as follows; first, a statistics gathering process in which the physical location i.e. the directory/filename of the file which contains the document(s) is stored along with the starting and ending location of every document within that file. Secondly, a document pre-processing procedure is carried out on identified documents. Thirdly, a partial index creation process is carried out on the data generated from the previous process. Fourthly, a partial index merging procedure is carried out in order to combine all of the partial inverted indexes into one complete inverted file index. Finally, a posting list ordering procedure is carried out on the complete inverted file to ensure that all of the posting list entries are keyed by a particular order.

5.2.1 Statistics Gathering.

The first stage in this transformation process is the identification and storage of all the necessary details in order to uniquely identify each document being indexed. The information stored for each document is as follows:

- Directory / filename containing the document.
- Name of the document.
- Starting location of document text within the file (byte offset from start of file).
- Ending location of document text within the file (byte offset from start of file).

- Number of index terms / phrases within the file.

This statistics gathering process is necessary to allow the document pre-processing phase to operate as efficiently as possible by identifying the starting and ending offsets of the indexable text within the document. This allows the document pre-processing phase to operate only on the information being transferred into the index and ignore the heading information contained in each document. Due to the way the *TREC* collection is distributed (as a large number of separate files each containing a number of documents), the statistics gathering process operates via a file at a time approach, i.e. it identifies and stores all necessary information about all documents within each *TREC* file passed to it. The *TREC* collection tags its documents using SGML tags with each document being in the following format:

```
<DOC>
        Heading information....
<TEXT>
        Document Text....
</TEXT>
</DOC>
```

This global formatting within the *TREC* collection allows us to efficiently identify all documents within the collection. The algorithm for this procedure is as follows:

```
START
while( end-of-file( ) == FALSE )
{
    doc-term-count = 0;
    doc-start = find-start-of-document( );
    doc-name = get-document-name( );
    find-start-of-text-within-document( );
    while( find-end-of-document( ) == FALSE )
    {
        term = get-term-from-document( );
        if( is-a-stopword( term ) == FALSE )
        {
            doc-term-count++;
            if( is-start-of-phrase( term ) == TRUE )
            {
                if( identify-phrase( term ) == TRUE )
                 doc-term-count++;
            }
        }
    }
    doc-end = current-position-in-file();
    write-doc-stats( doc-name, doc-start, doc-end, doc-term-count );
}
END
```

Figure 5.1 - Document Statistics Gathering Procedure

The net result of this procedure is the creation of the text collection's metadata, i.e. the information about the information being indexed. This metadata takes the form of one record for every unit of retrieval (be that a document or a document fragment) within the text collection.

5.2.1.1 Passage Level Retrieval.

As outlined in Chapter 3, up until recently the document was considered to be the smallest unit of retrieval the system could return, i.e. the system returned entire documents to the user in response to their request. In Chapter 3 research into the area of passage level retrieval was described. The overall consensus from this research [Buckley *et al* 1994] [Callan 1994] [Schäuble & Mittendorf 1994] and [Wilkinson 1994] is that passage level retrieval has beneficial effects on IR system effectiveness by allowing more fine-grained retrieval. It does however incur certain implementation overheads, but with careful selection of the approach used to implement passage level retrieval these implementation overheads can be managed efficiently in order to yield an improvement in effectiveness with minimal impact on efficiency. Passage level retrieval allows for the situation where the document as a whole is not very relevant to the query but certain passages or sub-sections within the document may be.

We decided to use a windowed approach to implement passage level retrieval similar to that described in [Buckley *et al* 1994] and [Callan 1994] which delimit passages by a count of the number of index terms as this facilitated greater control over the passage size and also the number of possible units of retrieval allowed. Our system was designed to handle overlapping and non-overlapping text windows of up to *Nw* index terms, with *Nw* being supplied to the indexing scheme. Using an overlapping approach pages overlap by half of the upper bound restriction on the number of index terms within the passage. For example, given an upper limit on the passage size of 200 words, then the first passage would start at the first indexable word in the document and continue until the 200th. The next passage would start at the 101st indexable word and continue until the 300th indexable word, and so on. The reasoning behind this

approach is to ensure that all concepts[10] with a length of less than 100 words are completely contained within one passage. This has a softening effect on the harsh delimitation imposed by the non-overlapping passages approach.

In the approach described by Callan, passages are generated on the fly during the retrieval process incurring a 25% increase in the implementation overhead in term of CPU time taken to process a query. We decided to design our system to avoid this overhead by splitting the documents into passages during the indexing process which is an off-line operation thus reducing the overhead during the retrieval process. This does mean reduced flexibility during the retrieval process however we felt that careful selection of the parameters of the document fragmentation process would eliminate the need for this flexibility.

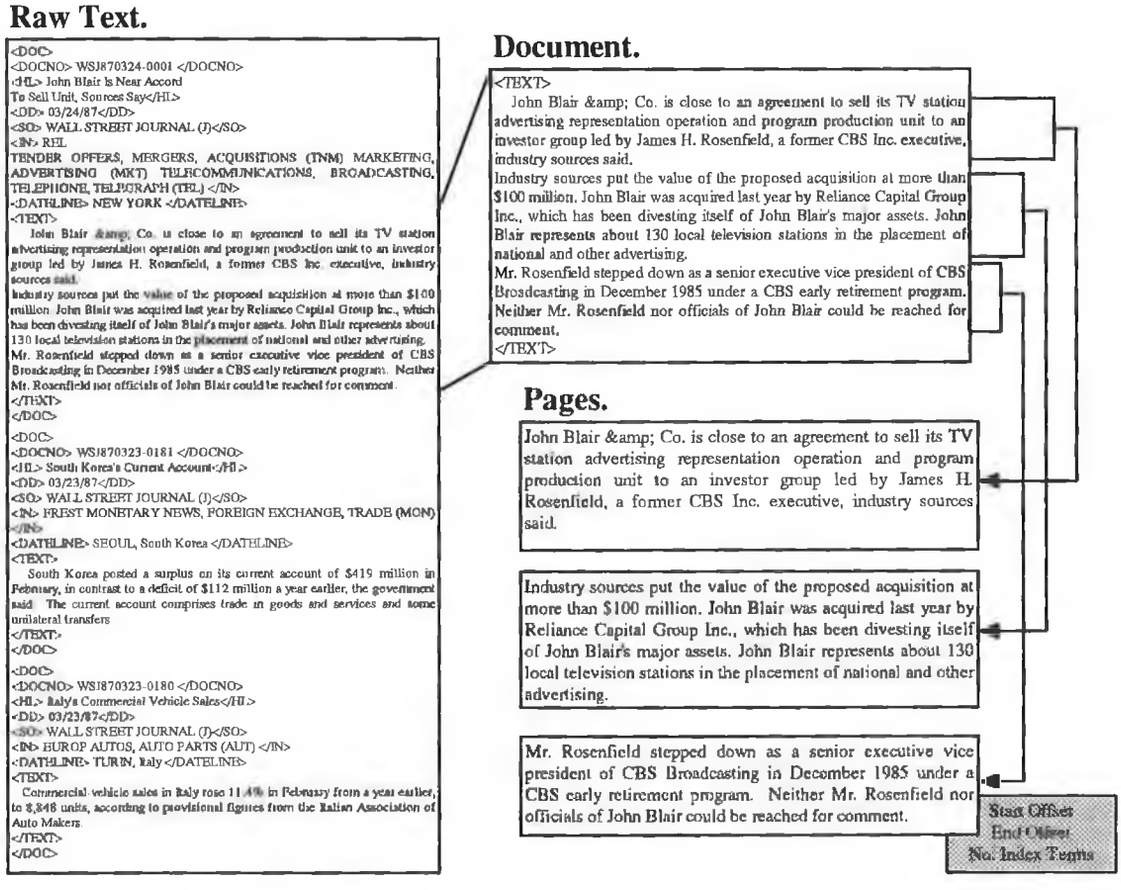


Figure 5.2 - Overview of the Paging process

[10] Section of text about a particular topic.

Figure 5.2 illustrates the general operation of the process which converts the raw textual information into smaller logical units of retrieval, in this instance passages which are non-overlapping text windows. Implementing passage level retrieval incurs some additional overheads, firstly, the actual cost of the paging operation itself. Secondly, the fact that paging results in a larger number of possible units of relevance[11]. Thirdly, the cost during retrieval of performing a passage to document resolution i.e. combining query-passage similarities to form query-document similarities. This resolution procedure is necessary in this instance because *TREC* relevance assessments are made upon whole documents not passages. Of the above three overheads the second and third present the most problems for the efficiency of retrieval engines as the first overhead is a pre-compute process and carried out off-line.

It is at this point in the indexing procedure when the passage delimitation process and the page size become critical to the efficient performance of an IR system. The smaller the passage size the greater the number of possible units of relevance. Care needs to be exercised in the selection of passage delimitation procedures whether based simply on counting keywords or more sophisticated approaches like the one described in [Schäuble & Mittendorf 1994].

For each document fragment, its associated document identifier is stored along with its physical location. Physical location in our test environment is its file number and its starting and ending offsets within the file. The number of index terms within the passage is also stored. These passage statistics are stored separately for use during the indexing operation and in the passage to document resolution procedure during retrieval.

Figure 5.3 illustrates the operation of the passage level statistics gathering procedure. When the number of index terms counted so far within a document exceeds a maximum value then a passage record is written out containing the starting location,

---

[11] Unit of relevance being the smallest object returned to the user by the search procedure be that whole documents or just document fragments.

ending location and number of index terms within the passage. The above statistics gathering algorithm has been modified to facilitate document fragmentation.

```
START
while( end-of-file( ) == FALSE )
{
        page-start = find-start-of-document( );
        doc-name = get-document-name( );
        find-start-of-text-within-document( );
        while( find-end-of-document( ) == FALSE )
        {
                page-term-count = 0;
                while( page-term-count < MAX-PAGE-SIZE )
                {
                        term = get-term-from-document();
                        if( is-a-stopword( term ) == FALSE )
                        {
                                page-term-count++;
                                if( is-start-of-a-phrase( term ) == TRUE )
                                {
                                        if( identify-phrase( term ) == TRUE )
                                        page-term-count++;
                                }
                        }
                }
                passage-end = current-position-in-file();
                write-page-stats( doc-name, page-start, page-end, page-term-count );
                page-start = page-end + 1;
                page-term-count = 0;
        }
        if( page-term-count > 0 )
        {
                page-end = current-position-in-file();
                write-page-stats( doc-name, page-start, page-end, page-term-count );
        }
}
END
```

Figure 5.3 - Passage Statistics Gathering Procedure

This procedure is only run once per collection or when a new file is added to the collection and all of the documents within the new file must be identified. It must be noted that all passages belonging to the same document will have the same document name. This is to facilitate the combination of query-passage similarity scores into a query-document similarity score.

5.2.2 Pre-Processing

This procedure reads the metadata one record at a time and processes the metadata records' corresponding document text. The raw text is read into memory and parsed. This parsing procedure removes all stopwords from the raw text, stems the remaining non-stopwords and then identifies phrases within the raw text.

5.2.2.1 Stopword Elimination.

Stopword removal has been shown to provide a number of benefits to the retrieval process. Improvements in efficiency are achieved due to the reduction in the number of index terms under consideration and hence the reduction in index size. Stopwords by their very nature occur very frequently within the text of the collection. Because the posting list length is directly related to an index term's occurrence frequency, stopword postings lists take up a significant amount of space within an inverted index. This means that the elimination of even a small number of stopwords and their corresponding postings will significantly reduce the overall index size.

Improvements in effectiveness are achieved by removing non-discriminating index terms from consideration during the retrieval process therefore considerably reducing the amount of index term noise in the retrieval process. Stopwords, because of their high occurrence frequency are of very little use in discriminating between one document and another because most documents will contain the same set of stopwords. Stopwords, therefore, would effectively contribute the same query-document similarity scores to all documents under consideration, thus yielding very little in terms of discriminating power and slowing down the entire retrieval process by having to process their posting lists.

Our stop list was constructed by initially taking the standard stop list from [van Rijsbergen 1979] and adding to it where we deemed necessary. Additions to the basic stopword list involved automatically including a number of high frequency terms, particular to our text collection. For example, *'Document'*, *'Wall St. Journal'*, etc.

On the surface, the stopword elimination procedure seems like a relatively simple one. However in today's IR environment where text collections are frequently in the multi-gigabyte range it can be seen that the stopwords must be removed from the raw text stream as efficiently and as early as possible so as to keep the indexing speed of an IR systems as fast as possible. The usual solutions to this problem are adequate, including binary trees, binary search of arrays, and hashing, with hashing being the fastest of the above. When hashing is used to search a stoplist, the stopword list must first be inserted into the hash table. Each incoming document token is then hashed into

the table. If the resulting location is empty, then the token is not a stopword and is passed on; otherwise, comparisons must be made to determine whether the hashed value really matches the entries at that hash table location. If there is no match, then the token is passed on but if there is then the token is a stopword and is eliminated from the token stream. This strategy is fast, but is slowed down by the need to re-examine each character in a token to generate its hash value and by the need to resolve collisions in the hash table. Although hashing is an excellent approach, an even better approach is possible, that is the removal of stopwords as part of the lexical analysis process. Since lexical analysis is carried out anyway as part of the indexing procedure, recognising even a large stoplist can be done at almost no extra cost during lexical analysis. This approach is extremely efficient. The overall procedure is for the lexical analyser generator to take as input the stoplist and to create a DFA (Deterministic Finite-State Automata). Figure 5.4 (taken from [Frakes & Bazea-Yates 1992]) illustrates the general idea behind the process for a small stoplist containing the following words (a, an, and, in, into, to):

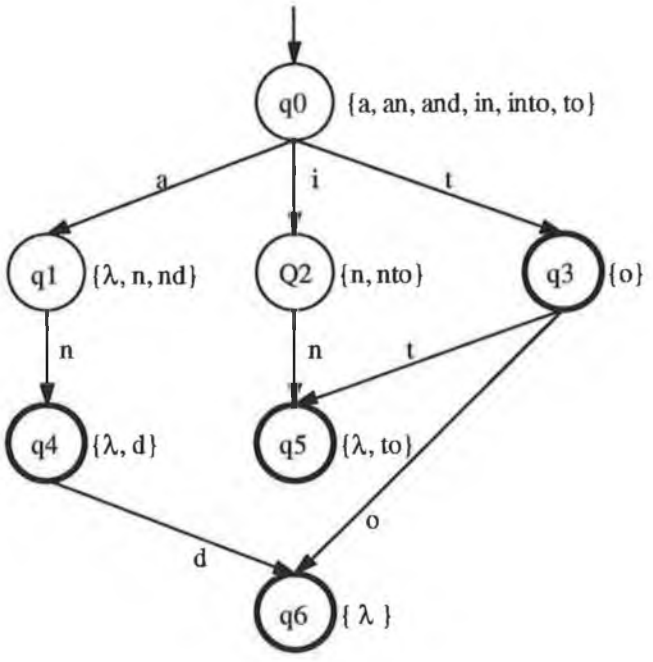


Figure 5.4 - DFA created for a stoplist.

The same procedure is used for creating the DFA for the full stoplist after which all of the text (both query and document) is passed through this filter. The net result of which is a list of indexable terms on which further processing may be carried out. For

all our experiments we used a stoplist of 410 words. This stoplist was formed by starting with a standard stoplist (taken from [van Rijsbergen 1979]) and augmenting it with the most frequently occurring words in the *TREC* collection.

5.2.2.2 Phrase Recognition.

A major bottleneck to the efficient operation of IR systems is the necessity to process terms of a query which contribute little in terms of discrimination value and yet consume a large proportion of the processing overhead during retrieval. The standard approach to this problem is to create a stoplist of words [van Rijsbergen 1979] which are discarded during the indexing and retrieval processes. This stoplist would include words such as 'and', 'but', 'they', 'which' etc. These words occur very frequently in normal text and as such are not useful in distinguishing one document from another with respect to a given query. Once these stopwords have been removed one is left with a reduced vocabulary of words or word stems which should provide an acceptable amount of term discrimination power.

However this is not always the case. Even after stopwords have been removed there still exist a large proportion of terms in the lexicon which occur in large percentages of the documents in the text collection, for example the terms 'bank' and 'computer' occur frequently within the *TREC* collection. One could argue that these frequently occurring terms could also be treated as stopwords and discarded from further consideration by the IR system. This in our opinion, is not an acceptable solution to the problem as this would cause the IR system to 'fail' in response to user queries containing such frequently occurring terms.

While the system's response to such user queries (short queries with commonly occurring terms) may not be very good it is still preferable to the system returning an error stating that it cannot proceed with the query because the terms entered were too general to be contained within the index's lexicon.

Our solution to this lexical generality problem is to expand the lexicon rather than restrict it. This expansion involves the recognition of commonly occurring phrases within the text collection and treating these phrases as single entities within the IR

system. It has been shown [Buckley *et al* 1994] that phrases are good document discriminators and the use of phrases yield performance improvements in terms of effectiveness.

5.2.2.3 Definition of a Phrase.

In order to extract phrases from text we must have a clear idea what exactly qualifies as a phrase. Within our test environment we have developed and use a phrase recognition procedure in which we regard any commonly occurring sequence of terms as a possible phrase. There are a number of restrictions to this rule. Firstly the phrase cannot begin with a stopword. This means that phrases such as '*the car*' and '*a house*' are not valid phrases. Secondly, phrases cannot end with a stopword. This eliminates phrases such as '*buy a*' and '*play the*'. Thirdly, phrases can contain stopwords, this allows phrases such as '*department of defence*' and '*passing the buck*'. Another criterion for a term's inclusion into a phrase is the occurrence of consecutive terms all of which start with capital letters. This criteria is included to aid the recognition of commonly occurring names of people e.g. '*George Bush*' , company names, e.g. '*International Business Machines*', and place names, e.g., '*New York*' and '*San Francisco*'. We must also formally define what constitutes 'commonly occurring' phrases, i.e. how frequently term co-occurrences must occur in order for them to be classified as phrases.

5.2.2.4 Phrase Extraction.

Due to the way our IR system is developed the phrase extraction procedure was easily implemented by taking an existing document pre-processing module of our IR system and modifying it. The easiest way to explain its operation is by example. Take the following extract from the *TREC* text collection:

> *The celluloid torch has been passed to a* ***new generation****: filmmakers who grew up in the 1960s. ``Platoon,'' ``Running on Empty,'' ``1969'' and ``Mississippi Burning'' are among the movies released in the past two years from writers and directors who brought their own experiences of that turbulent decade to the screen. ``The contemporaries of the '60s are some of the filmmakers of the '80s. It's natural,'' said* ***Robert Friedman****, the* ***senior vice president*** *of* ***world wide*** *advertising and publicity at* ***Warner Bros. Chris***

__Gerolmo__, who wrote the screenplay for "Mississippi Burning," noted that the sheer passage of time has allowed him and others to express their feelings about the decade. "Distance is important," he said. "I believe there's a lot of thinking about that time and America in general." The __Vietnam War__ was a defining experience for many people in the '60s, shattering the consensus that the __United States__ had a right, even a __moral duty__ to intervene in conflicts __around the world__. Even today, politicians talk disparagingly of the "__Vietnam Syndrome__" in referring to the country's reluctance to use __military force__ to settle disputes.

The bolded portions of the text extract are possible candidates for classification as phrases, they include frequently occurring phrases such as '*new generation*', '*moral duty*' and '*around the world*', people's names '*Robert Friedman*' and '*Chris Gerolmo*'. Word co-occurrences like '*celluloid torch*' would not be treated as a phrase due to relatively infrequent co-occurrence of the individual terms in the document set.

It must be remembered that meaningful phrases cannot be extracted just from this extract of text alone. The phrase recognition process only becomes effective when a large amount of textual information is processed. This allows statistical information to be gathered on the frequency of occurrence of phrases. The following figure illustrates the method used to extract phrases from the document text:

W1	W2	W3	Output
been	passed	to	None
passed	to	a	None
to	a	new	None
a	new	generation	None
new	**generation**	filmmakers	Yes

Figure 5.5 - Phrase Extraction from Text.

It must be noted that our phrase recognition process automatically detects phrases of length up to and including three terms. A sliding window limited to a width of three words moves through the document text when the window begins with a stop word, then that word is skipped and the contents of the window are shifted left by one position. If the first word is a non stopword then output is produced only if the ending term is also not a stopword. Once a candidate phrase has been located it is stored and if this is the first occurrence of that candidate phrase then a new storage structure is allocated to it and the candidate phrase's document identifier is also stored.

Once all of the documents have been processed the output is a list of candidate phrases and all of the documents they occur in. This list is then sorted by phrase and a count of the number of unique document identifiers in which the phrase occurs in is taken. It is this count which determines whether or not the candidate phrase is included into the phrase set for the collection.

In our implementation the recognition process involves processing document text in sections, with each section being around 20 Mbytes. This section size is determined by the amount of memory available for the process on the machine. For each section of text, a file containing each possible candidate phrase along with its occurrence frequency is produced. This file is then sorted by decreasing occurrence frequency and all phrases with an occurrence frequency of greater than 25 (i.e. the phrase occurs in more than 25 different documents within the current text section) is included in the phrase set. The selected phrases are then added to a global phrase set.

The phrase extraction process is a once off event for static text collections and a periodic one for dynamic text collections. The stopping criterion for the phrase extraction procedure can be one of two conditions. Firstly, the process is repeated until the number of new phrases being added to the global phrase set falls below a certain threshold value. At this point it can be assumed that the vast majority of the phrases have been located and extracted. This assumption depends on the text collection being static in nature. If the IR system was dealing with a dynamic text collection then the phrase recognition procedure would have to be run periodically when the amount of new documents added to the collection allowed the statistical extraction of new phrases. Secondly, it is by no means computationally prohibitive to apply the procedure to the whole text collection.

Applying this phrase recognition approach to the *TREC* collection resulted in the identification of 219,770 commonly occurring phrases within the collection itself. The phrases generated from this approach have a distinct advantage over a pre-defined phrase set extracted from a third party source in that they are extracted from the text collection being indexed therefore the number of phrase matches attained using these phrases will be much higher. This however does not limit their use to this text collection only. Once generated the phrases can be applied to any text collection. It

must also be noted that all component terms that make up the phrases have already been passed through the stemming process, this means that phrases like '*computer department*' and '*computing departments*' will be reduced to their base form '*comput depart*'. This has the effect of increasing the probability of matches between phrases and terms within the documents.

5.2.2.5 Stemming.

The non-stopwords and phrases identified from documents are passed onto the stemming and conflation procedure. Our stemming procedure was used to reduce terms to their word stem. This can be accomplished algorithmically [Porter 1980], via exception lists, or a combination of the two (as done in WordNet). Initially we used the WordNet stemmer along with its exception lists. However our experience has shown that this stemming approach incurred a relatively high overhead during the indexing process when compared to the purely algorithmic approach of Porter's stemming. We then switched our stemming procedure to Porter's stemming algorithm which resulted in better performance in terms of efficiency because of the elimination of the need to store the stemming exception lists in memory and the elimination of the need to search these exception lists for every word being stemmed. The output from this procedure is a list of stemmed index terms and collocations, for example, words such as *'computer'*, *'computation'*, *'computerise'* would be conflated to their common word stem *'comput'*.

The inclusion of any stemming mechanism has implications for efficiency and effectiveness. Efficiency is improved by reducing the number of unique index terms under consideration as illustrated in the previous example therefore eliminating the need for and overhead of separate posting lists for each non-stemmed index term. This further reduces the size of the index and hence speeds up access. Effectiveness improvements also result from the use of stemming due to the remaining stemmed index terms being normalised. To use the previous example, a query containing the term *'computerise'* stemmed to *'comput'* will match documents with the terms *'computer'* and *'computation'*. This has the effect of drawing in documents into the retrieval net which would otherwise be ignored.

5.2.3 Internal Document Representation.

Discriminating words and phrases from the tokenisation of a document are passed to a procedure which stores them in such a fashion so as to facilitate the easy creation of a partial inverted file index. This index term storage procedure creates a binary insert tree in main memory. This index term binary tree contains a node for each unique index term. Each node in the index term binary tree has a pointer to another binary insert tree which contains the posting information for every occurrence of the index term. Each node in this posting list binary tree contains the document identifier the corresponding index term occurred in along with its occurrence frequency within the document.

The net result of this procedure after it has processed a number of files each containing a number of documents, is an index term binary tree with each node in the tree representing a unique index term. Associated with each node in this index term binary tree is a pointer to another binary tree, which holds the posting and positional information for each unique index term. This is graphically illustrated below:

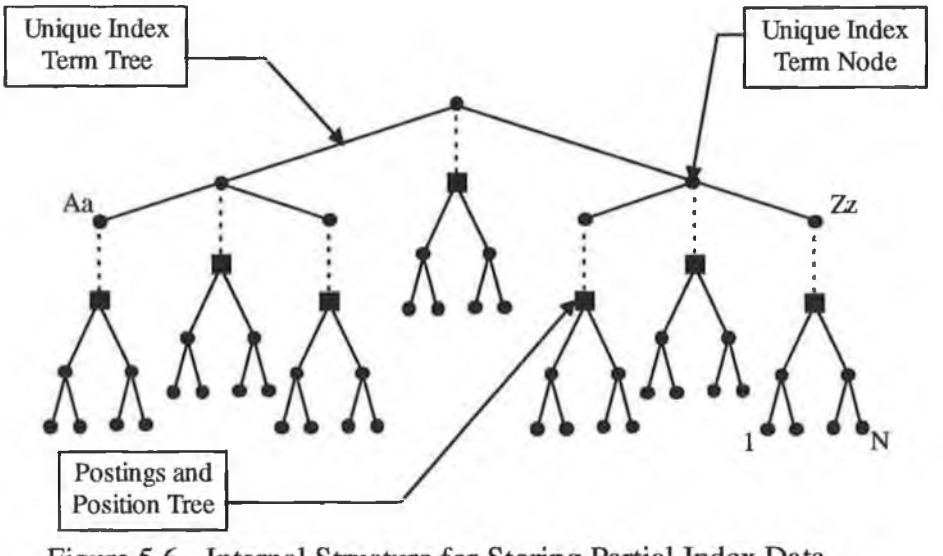


Figure 5.6 - Internal Structure for Storing Partial Index Data.

Each unique index term node in the index term tree holds a certain amount of information about each unique index term, this information is as follows:

- A string representation of the unique index term itself.
- The number of documents the index term occurs in.

- A pointer to another binary tree structure (which holds the posting and positional data for that index term).
- Right and left pointers to other nodes in the unique index term tree structure.

Each node in the posting and positional tree structure holds the posting and positional data; this information is as follows:

- The unique-document identifier.
- The within document frequency of the index term in the document (how many times it occurs in the document).
- The first occurrence position of the index term in the document.
- The last occurrence position of the term in the document.
- The average occurrence position of the index term in the document.
- Right and left pointers to other nodes in the posting and positional tree structure.

This tree building process continues until the physical memory of the computer is exhausted. The program could continue and use the virtual memory of the machine but this becomes inefficient as a lot of unnecessary I/O due to page swapping will then occur. Once the maximum memory limit has been reached the index term binary tree is passed to a function which performs a depth first search of the tree. For each node processed in the index term binary tree a further depth first search of the index term's posting list binary tree is also carried out. In this manner the data required to create a partial inverted index is generated. The data generated is in the following format:

Lexicon Data		Posting Data		Position Data		
Index Term	**No. of Postings**	**Doc. Id**	**Term Freq.**	**Min Position**	**Max Position**	**Average Position**
Aaron	3	200	4	145	178	161
		234	3	10	677	213
		567	8	121	144	132
Aeroplane	2	45	10	3	990	150
		67	12	54	567	234
.		.		.		
.		.		.		
.		.		.		
Zoom	4	90	2	50	100	75
		120	4	132	167	146
		150	5	877	951	921
		178	1	144	144	144

Figure 5.7 - Internal Document Representation.

The internal representation of the text being indexed is illustrated in Figure 5.7. It consists of three main areas, firstly the lexical information, secondly, the posting information and thirdly, the positional information. The lexical information contains a record for every unique index term in the text being processed along with the number of documents that unique index term occurs in. The posting information contains a record for every document that each unique index terms occurs in, along with the frequency of that index term within all documents. The positional information records for every document that each unique index term occurs in, statistical position information namely the first, last and average occurrence positions of the index term in the document (measured in terms of a byte offset from the start of the document). It must be noted that the average occurrence position must be computed and stored at this point in the process because the information needed to compute the average occurrence position is discarded after it is computed.

5.2.4 Statistical Position Information.

As illustrated in Figure 5.7 a certain amount of term position information is calculated and stored during the generation of the internal document representation. The incorporation of positional information into an index used by an IR system allows more complex retrieval operations to be carried out by the system in order to improve system effectiveness. These additional retrieval operations would include things like term to term proximity calculations which could result in the document's overall similarity score being modified to reflect the closeness of the co-occurring query terms in the document. For example, a document whose accumulated query document similarity score is derived from three partial query document similarity scores from three query terms occurring within the document but far apart from each other would probably not be as relevant as another document which contained the same three terms in close proximity to each other. The downside of incorporating positional information into an index is a large increase in the overall size of the index and just as importantly yet another level of complexity added to the structure of the index itself. An inverted index itself by its very nature is difficult to handle due to the fact that it is composed of variable length records. Incorporation of positional information leads to each posting entry becoming variable in length as illustrated in Figure 5.8.

No Positional Information.			
Term	**No. Postings**	**Doc. Id.**	**Within Doc. Freq**
comput	3	45	1
		56	3
		78	2

Positional Information Included.							
Term	**No. Postings**	**Doc. Id.**	**Within Doc. Freq,**	**Pos1**	**Pos2**	**...**	**PosN**
comput	3	45	1	124			
		56	3	13	45	56	
		78	2	689	1002		

Figure 5.8 - Effect of including Positional Information.

This results in variable length posting entries within variable length posting lists which are computationally expensive to update and maintain efficiently. However in spite of the obvious difficulties in handling positional information in an index, the potential for improved effectiveness by using this positional data is very strong. In order to try and exploit the advantage of using positional information during retrieval while maintaining a high degree of efficiency, we developed during the course of our research a method for storing positional information in a fixed length format therefore eliminating the need for the introduction of a second level of record variability into the index structure.

The existing posting information stored by our system is the unique document identifier n where $1 \leq n \leq N$ and the within document frequency (WDF) of the term in the document. This information is stored (using a simple compression method) in one unsigned long (4 bytes). In order to maintain as much efficiency as possible by keeping the index structure as simple as possible it was decided that whatever positional information was stored must fit into the same amount of space (4 bytes). The reasoning behind this approach was to create a positional information file which has an identical structure to the postings file but contains positional information instead of posting information. This is illustrated in Figure 5.9.

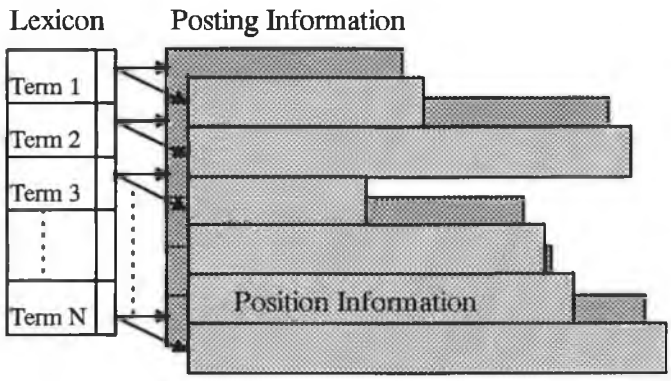


Figure 5.9 - Inverted File Index Containing Positional Information.

Due to the fact that the positional information file mirrors the structure of the posting information file exactly, the posting and positional information for a given query term document pair are at exactly the same position in both files. This means that no modifications have to be made to the structure of the lexicon and no additional information need be stored in the lexicon because the same offset is used for both the posting and position files. The complete separation of the posting and positional data means that the use of the positional data can be switched off if wanted and no overhead is incurred in skipping the positional data as would be the case if the positional information was embedded in the postings file. As illustrated in Figure 5.7 the positional information that is stored is the occurrence position of the first instance of the term in the document, the occurrence position of the last instance of the term in the document and the average occurrence position of all the instances of the term in the document. These values are stored as byte offset values from the beginning of the document. Obviously there exists a problem in storing three term bytes offset values in one unsigned long (4 bytes) when a document or document fragment can be of any length. Our approach to solving this problem was to firstly, compute the three bytes offset values (Min, Max and Average) for each unique term occurring with the document or document fragment and then divide each of these values by the length of the document or document fragment in bytes, this gives us three numbers in the range 0 to 1 representing the first, last and average positions. These numbers were then scaled up to integers in the range 1 to 999. These integer values represent an estimation of the position of the first, last and average positions of the term in the document. There is a certain amount of error associated with each value with the error depending on the length of the document. The longer the document the greater the

amount of error between the estimated position and the actual position. However since our system uses passage level retrieval instead of document level retrieval, the positional offsets are taken from the start of the passage rather than the start of the document, passage sizes are strictly controlled and are an order of magnitude smaller than the original document sizes. This in effect means that the error incurred by this approach is minimised.

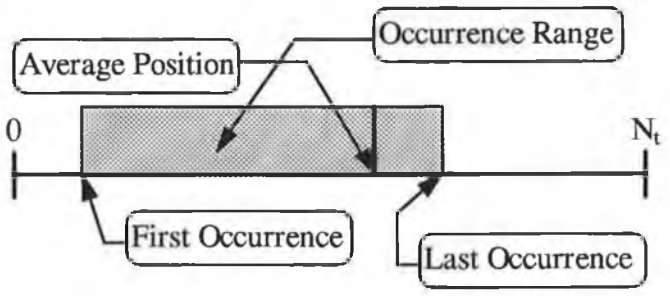


Figure 5.10 - Graphical Representation of Positional Information.

Figure 5.10 illustrates graphically the positional information stored in the modified index with the document or document fragment being represented by a vector of 0 to N_t terms. The positional information stored can be used to define a range or subsection of the document or document fragment in which its associated term occurs in. Naturally if the term only occurs once within the document the first, last and average occurrence positions will be the same, if the term only occurs twice then the minimum and maximum positional will be the only part of the positional information of value. It is only when the within document occurrence frequency is greater than two does the average occurrence position give us some idea of the occurrence distribution of the term within its minimum and maximum bounds.

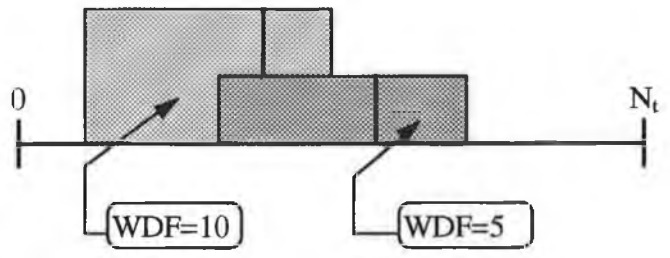


Figure 5.11 - Combination of Positional Information.

Positional information from all query terms occurring within a document can be combined and overlaid to give a graphical representation of the occurrence of the

terms within the document as illustrated in Figure 5.11. The graphic representation of the document of length N_t words in Figure 5.11 has two query terms occurring in it, one which occurs ten times in the first half of the document fragment and the other which occurs roughly in the middle third of the document fragment. In our system we use query expansion as explained in Section 5.3.6. This procedure automatically selects text from the top N_d documents from the ranking the system returned in response to the initial query. Our primary focus in developing this positional information was to enhance the automatic query expansion procedure incorporated in our retrieval engine by restricting the amount of text fed back into the query expansion procedure from the automatically selected top N_d documents from the output of the initial query. The positional information values from all the query term occurring in the document fragment can be overlaid and a cumulative positional graph generated as illustrated in Figure 5.12, this information can then be used to select subsections of text from the document fragment to be fed into the query expansion procedure.

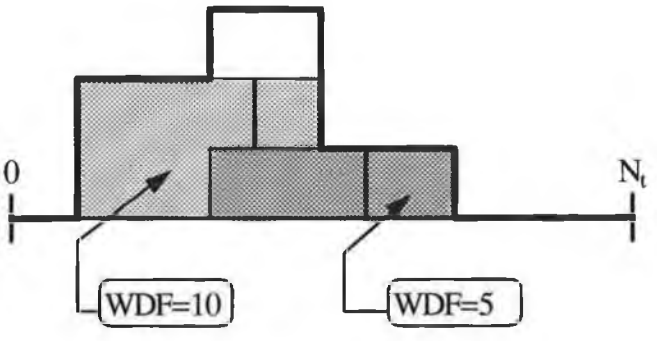


Figure 5.12 - Cumulative Positional Information Graph.

The bold line in the graph represents the cumulative positional weight of all the query terms at each point throughout the document. It can be clearly seen even from this simple example that there exists a section within the document where occurrences of the two query terms overlap. It is probable that that portion of the document fragment would be the most relevant part of the document fragment with respect to the query. This approach can be further modified to reflect the index term's weight in the context of the current query. The height of the shaded areas in the above diagrams would then represent the query term's weight in the context of the current query and not just the 'within document frequency' of the term within the document.

Another use of the positional information stored in the index would be to generate an iconic representation of the document and display this document icon beside the document name in the ranked list returned to the user. The document icon would be made of the document fragment icons concatenated together. This would give the user of the IR system an immediate idea of the distribution of the query terms within the document especially if the document was long. We call this type of visualisation *'Document at a glance'* (DAAG).

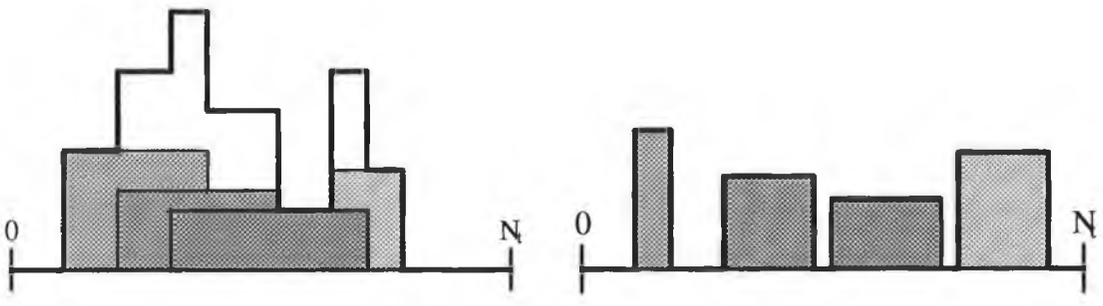


Figure 5.13 - Iconic Representation of Documents.

Figure 5.13 illustrates graphically the positional distribution of the query terms within two documents. If the user was presented with these graphs as icons they would immediately be able to determine that both documents contain the same query terms but that in the first document the query terms overlap within the document to a considerable degree while in the second document no overlap exists between the query terms in the document. Even greater effectiveness would be obtained if colour was used instead of grey shading as we illustrated in Figure 5.13. This would indicate that the first document is probably more relevant to the query than the second document.

The closest contemporary visualisation technique for query term occurrences in variable length documents is called 'Tilebars' [Hearst *et al* 1995]. While this statistical positional information is supported in the design of our IR system and is scheduled for further research in the coming year the central focus of this thesis is on the optimisation of the retrieval process.

5.2.5 Partial Inverted Index Creation.

The data generated during the pre-processing stage of document indexing is passed to a function which transforms it into an inverted file. This inverted file consists

of three distinct parts, the lexicon, the postings file and the position file. The lexicon contains one record for every unique index term contained in the set of documents being indexed. Its structure is as follows:

- Index Term.
- Number of Postings.
- Maximum within-document term frequency.
- Posting List Offset.

The index term contains the actual index term itself, in our experimental environment we impose a maximum size on the index term of 48 characters. This upper bound was derived in order to accommodate phrases of up to 6 component terms. The number of postings contains the number of unique documents the index term occurs in. The maximum within-document term frequency holds the maximum number of times the index term occurs within one document. This value is stored in the lexicon to allow ease of access during the search procedure. It eliminates the need to access the posting list in order to determine the maximum within-document term frequency. The posting list offset is simply a byte offset into the posting list file and indicates exactly where the index term's posting list starts. After moving to this offset position in the file all that is then required is to read in the index term's number of postings.

For each unique index term there are a number of postings associated with it (specified by the number of postings value in the index terms lexicon structure). The structure of a posting is as follows:

- Unique document identifier.
- Within document term frequency.

This posting structure is stored in a compressed form of one unsigned long (4 bytes). The compression procedure is relatively simple. The maximum number that can be stored in an unsigned long is 2^{32}-1 which is 4,294,967,295. Within our experimental environment (the *TREC* text collection) there are just over one million documents, which leaves us with plenty of room for storing the additional within-document term frequency information within the same unsigned long as the unique document

identifier. The unique document identifier is simply shifted left by three decimal positions. This allows for a maximum within-document term frequency of 999. Initially this may seem like a small enough limit to impose on the within-document term frequency but when incorporated with the document fragmentation procedure described above in which the page size would never be allowed to exceed this value, the problem is removed.

The net result of this procedure is the creation of a partial inverted index representing a portion of the collection being indexed. The size of this portion is limited to the amount of memory available on the indexing computer.

5.2.6 Partial Inverted Index Merging

Once all documents in the text collection have been indexed and their respective partial inverted files created, these partial inverted files must then be merged into one overall inverted index. The inverted index merging procedure is similar to any standard merging procedure. The algorithm is as follows:

```
START:
no-of-files-to-merge = get-inverted-file-count( );
while( no-of-files-to-merge > 1  )
{
      if( no-of-files-to-merge Mod 2 )
      {
            i = no-of-files-to-merge;
            merge-inverted-files( i -1, i, i -1 );
            no-of-files-to-merge = no-of-files-to-merge -1;
      }
      j = 0;
      for( i =1;i < no-of-files-to-merge; i = i +1 )
      {
            merge-inverted-files( i, i+1,j++ );
      }
      no-of-files-to-merge = j -1;
}
END:
```

Figure 5.14 - Partial Index Merging Procedure

The actual merging procedure takes two partial inverted indexed and merges them to create a third partial inverted index. Matching records in the two input partial inverted indexes have their posting lists concatenated together. It must be noted that this posting list concatenation procedure does not preserve the ordering of posting entries by document identifier. The posting list ordering procedure is carried out in the

inverted index post-processing phase. On completion of this merging procedure the partial inverted indexes have been converted into one overall inverted index.

5.2.7 Inverted Index Post-Processing.

Once all partial inverted indexes have been merged into one overall inverted index a certain amount of post-processing must be carried out on the index. This post-processing takes the form of eliminating all words which occur only once in the entire text collection. Such words are deemed to be either misspellings or once-off occurrences of company or place names and as such are very unlikely to be of relevance during retrieval. We have found that these once-off occurrences account for nearly half of the unique index terms in the lexicon. So applying this restriction reduces the size of the lexicon and therefore reduces the time taken to search it but the overall inverted index size is not greatly reduced because the posting lists associated with these once-off index term occurrences are very short.

The other action performed during the index post-processing phase is the sorting of the index term's posting lists on a key. In our experimental environment this key can be one of two things, firstly, increasing unique document identifier or secondly, decreasing within document term frequency divided by document length. The algorithm for the procedure is as follows:

```
START:
{
    open-old-inverted-index( );
    open-new-inverted-index( );
    while( end-of-old-index( ) == FALSE )
    {
        old-posting-offset = read-old-lexicon-entry( );
        if( current-posting-list-length > Threshold )
        {
            old-posting-list = read-posting-list( old-posting-offset );
            new-posting-list = sort-posting-list( old-posting-list );
            new-posting-offset = write-posting-list( new-posting-list );
            write-new-lexicon-entry(  new-posting-offset );
        }
    }
    delete-old-inverted-index( );
}
END:
```

Figure 5.15 - Index Post-Processing Procedure

Both approaches to sorting have their advantages and disadvantages. Sorting by unique document identifier has the advantage of making modifications to the inverted

index easier by allowing the easy insertion of new posting information into the posting list, i.e. because the posting lists are sorted in order of increasing unique document identifier as a new document's identifier will always be greater that any existing document identifier the new posting information is simply appended to the end of the posting list. Another advantage of this sorting method is that it allows the posting list to be run length encoded and compressed, thus reducing the overall size of the inverted index and the total I/O time required during retrieval. The disadvantage of this sorting approach is that while the posting lists are in this order it is impossible to apply thresholding techniques to the posting lists during the retrieval process. In order to implement posting list thresholding on a posting list sorted in this manner the entire posting list would have to be read into memory and then re-sorted by the decreasing within-document term frequency divided by document length key before any processing savings can be gained from thresholding.

Sorting the posting lists using within document term frequency divided by document length have the disadvantage of making the index slightly more difficult to update. Because the posting lists are not sorted by unique document identifier the insertion of a new posting requires a search through the existing posting list information to determine where the new posting should be inserted. However sorting the posting lists on this key has advantages during retrieval. Because the posting information is sorted in order of decreasing importance to its index term a dynamic run-time threshold can be imposed on each posting list depending on the input query. This has the effect of eliminating the need to retrieve the entire posting list into memory from disk and process it. Once the posting lists threshold has been determined then only that portion (up to the threshold) need be processed. This form of posting list sorting also facilitates more advanced forms of Query Space visualisation and modelling.

5.3 Retrieval

The document search procedure can be divided into a number of distinct phases. Firstly, the query must be converted into an acceptable internal format from which the IR procedure can begin. Secondly, a pre-computation phase is carried out in which all values necessary for retrieval are computed once and stored. Thirdly, an inverted file

access phase in which each relevant posting is accessed and processed. This results in query document similarity scores being generated. Fourthly, the normalisation and ranking of all query document similarity scores and lastly, the output of the ranked list of results.

One of the overriding objectives in our research was to keep the retrieval overheads even for large text collections as low as possible. With this criterion in mind we designed the retrieval aspect of our IR system. If this system is to be used in a multi-user environment where there are numerous concurrent accesses the memory overhead of the search engine must be kept to a minimum. The amount of disk I/O should also be kept to a minimum.

5.3.1 Query Pre-Parsing

Our system was designed to allow two types of interaction, interactive and batch. When used in interactive mode the system accepts query terms entered directly from the command line. Alternatively the search engine can be invoked in batch mode where any number of previously defined queries can be passed to the system for processing. Both forms of query are converted into the same internal representation.

The identification of phrases both in documents and queries is a non-trivial one and requires a significant amount of memory overhead in order to run efficiently. It is more efficient to eliminate the need to identify phrases within the query text. To this end we developed an approach which automatically generates all possible phrases from the query text by excluding the frequency of occurrence constraint which requires a minimal number of occurrences and then checks to see if they occur within the text collections lexicon. If they do then the phrase is added to the query and incorporated into the query's metadata structure. The procedure of generating a candidate phrase set and then matching it against the lexicon is more efficient that the procedure identifying a phrase that definitely occurs in the lexicon. This is due to the fact that the second approach requires that all of the phrase data be loaded into memory (in our case this equates to 219,770 phrases and the structure information necessary to hold them) while the first approach incurs very little additional overhead during retrieval. This inverted representation is then manipulated in such a fashion as to allow the most

efficient processing of the query. The internal representation of a query within our system is as follows:

No. Posting	Query Freq.	Max. Within Doc. Freq	IDF	Query Term Weight	Query Term Threshold	Posting List Threshold	Query Term
20	2	5	4.35673	23.56734	TRUE	20	inmat
22	1	6	4.45780	28.99431	TRUE	22	reliev
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
798	1	6	1.1247	1.56732	FALSE	349	countri

Figure 5.16 - Internal Representation of Query

The table above details what information is computed and stored about each valid index term extracted from the initial input query. The index term's metadata is made up of the following:

- The number of postings associated with the index term, i.e. the number of unique documents the index term occurs in.
- The frequency of occurrence of the index term in the query text.
- The maximum number of times the index term occurs within one document.
- The Inverse Document Frequency for the index term (a measure of the term's specificity).
- The query term weight is the overall weight assigned to the query term during processing.
- The Query Term Threshold flag for the index term indicates whether it will be processed if the *QTT* option is switched on.
- The Posting List Threshold value indicates the number of postings from the entire posting list that will be read in and processed.
- The index term string itself.

This information encodes the overall shape of the Query Space (QS) for a given query. This QS being the area of data actually processed during retrieval. Once all of this information is computed it is then sorted in order of increasing posting list length (also increasing IDF score). This means that the most specific terms (ones which occur in the least number of documents) are processed first.

5.3.2 Pre-Compute Phase

It is in the pre-compute phase where most of the above index term metadata values are computed. The IDF and QTW values can be computed here because we know a term's occurrence frequency within the collection along with the total number of documents in the collection. If the *QTT* and *PLT* options are switched on the additional flags and values are computed and set. The *QTT* procedure determines if the occurrence frequency of a term within the collection is too great to be of benefit during the processing of the query. If a given term is deemed to be too general in the context of the current query, a flag is set to reflect this and the index term is eliminated from further processing. The *PLT* procedure, when activated, determines the amount of the index term's posting list that is processed during retrieval. If this option is not switched on the default is to process all of the posting list otherwise the *PLT* threshold is computed as a percentage of the total number of postings in the index term posting list.

A certain portion of pre-computing must be left until each individual posting list is being processed, namely the calculation of the final query document similarity score components. Once the index term's query term weight and the maximum within-document frequency (MWDF) values are known then the values of QTW_{tf} with *tf* ranging from 1 to MWDF are computed. This eliminates the need for computing these values repeatedly in long posting lists.

The purpose of this phase is to eliminate as much of the repetitive computation of values as possible therefore saving time during the inverted file access phase. Any overheads incurred during this procedure will be more than offset during the inverted file access phase.

5.3.3 Inverted File Access

This phase processes the query terms metadata one record at a time. Up until this point in the retrieval process no major memory overheads have been incurred. However at this point we need some sort of structure (capable of being held within main memory) that has the ability to efficiently handle the accumulation of many query-document similarity scores. The structure we decided upon is a binary insertion tree

with each node in the tree representing a unique document in the collection. This tree starts off with a single root node and gradually expands as partial query document similarity scores are computed and inserted into the tree. If a unique document identifier does not already exist within the tree structure then a new node is generated and inserted into the correct position within the tree. If a node already exists then the new partial query document similarity score is added to the existing one. A count of the number of query terms which contributed to the total query document similarity score is also kept. This tree therefore incurs the minimum amount of overhead in that it only allocates what memory it needs and no more, i.e. there are no previously defined static arrays. The tree structure created during this process can be assumed to be roughly balanced because the information being inserted into the tree is presented to the insertion process in random document accumulator order.

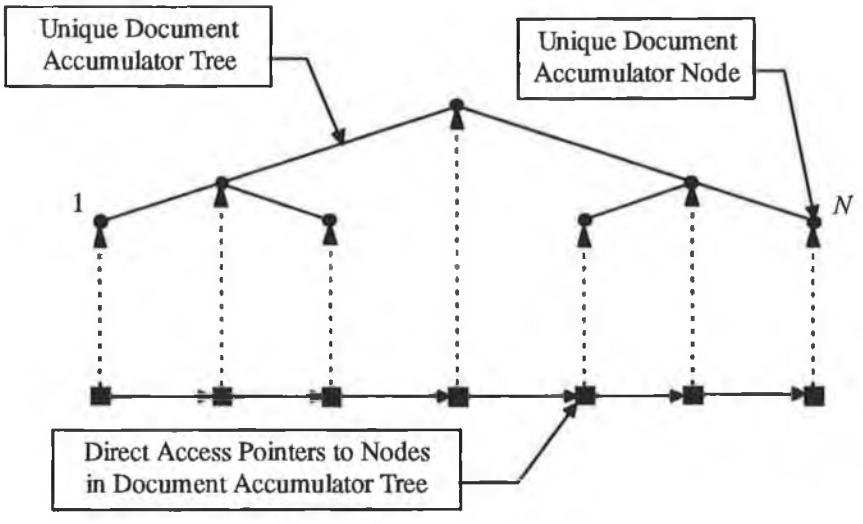


Figure 5.17 - Accumulator Tree Structure.

As illustrated in Figure 5.17 an additional set of pointers which are directly linked to each node in the binary insertion tree is also created. This additional set of pointers is needed in order to efficiently sort the binary tree from its original order of increasing unique document identifier to one of decreasing query-document similarity scores.

The inverted file access procedure works as illustrated in Figure 5.18. The query is processed one index term at a time, with the most discriminating index terms (those with the highest *IDF* scores) being processed first.

```
START
{
    while( end-of-query( ) == FALSE )
    {
        query-term = get-next-query-term( );
        posting-list = access-posting-list( query-term );
        while( end-of-posting-list( posting-list ) == FALSE )
        {
            current-posting = extract-posting( posting-list );
            process-posting( current-posting );
        }
    }
}
END
```

Figure 5.18 - Processing Inverted File Posting Lists.

As the algorithm described in Figure 5.18 proceeds, the accumulator tree structure is gradually built up. As each new node is inserted into the tree a pointer to this new node is stored in a linked list of pointers. This means that the memory required to hold the active accumulators is kept to a minimum. No statically defined structures are required which would tie up large amounts of core memory for long periods during retrieval.

5.3.4 Normalisation and Ranking

When dealing with a free text collection document lengths are generally not uniform. Even in situations where documents are all the same type as in newspaper articles or document abstracts, one finds a good deal of variability. This non-uniformity causes problems during the retrieval process as longer documents will naturally attain higher overall query document similarity scores simply because they contain more index terms than short documents. If no form of compensation for this is taken then the IR system would be biased towards longer documents. This is clearly unacceptable, so a normalisation procedure for document lengths is required.

Handling the bias can be achieved in one of two ways, firstly, by the incorporation of passage level retrieval techniques into the IR system (as described in Section 5.2.1.1), or secondly, by dividing the accumulated query document similarity score by some function of the document length, i.e. the log of the number of index terms within the document (the document length).

Normalising by the document length may still be necessary if passage level retrieval is incorporated into the IR system depending on how the passages are delimited. Some passage delimiting procedures, for example, [Hearst & Plaunt 1993], result in shorter but still variable length passages so the requirement for normalising by the passage weight still exists. Once similarity scores for all active document accumulators have been normalised they are passed to the sorting procedure.

The tree generated during the inverted list processing stage only contains the minimum set of document accumulators i.e. those which have attained a non-zero weight during the inverted list processing phase. Because the additional set of pointers directly access each document accumulator node within the tree, the sorting procedure can be carried out without moving any of the node information around. The swapping of two nodes is achieved through the swapping of the pointers to the nodes themselves. This has the effect of reducing the sorting overhead to a minimum.

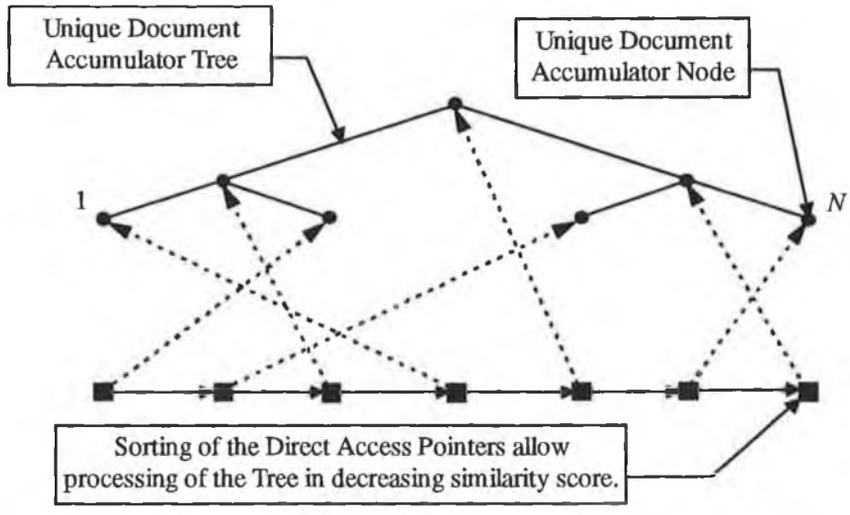


Figure 5.19 - Result of the Accumulator Sorting Procedure.

In most retrieval situations the number of document accumulators that attain weights during retrieval (N_a) is much greater than the number of documents actually returned to the user (R). This fact would suggest that a conventional sorting procedure applied to N_a accumulators and requires N_a Log(N_a) comparisons is inefficient if the number r is much smaller than N_a ($R << N_a$). However within our experimental environment the condition $r << N$ does not hold, rather a $R < N_a$ condition is the case. This is due to accumulator activation restrictions employed by our system during the

retrieval process (detailed in Chapter 6). Typically in our situation the sorting procedure is highly efficient and does not cause a bottleneck in the retrieval process.

5.3.5 Output of Results

Once all of the results have been accumulated, normalised and sorted they must be presented in rank order. A further normalisation of the scores produced during the retrieval process is applied to the scores of the top R documents to be returned to the user. This normalisation results in the scores for the top R documents being in the range 0 to 1, with 1 being the highest scored. The information produced is a simple list containing the unique document identifier and the document score.

5.3.6 Automatic Query Expansion.

Query Expansion (QE) is one approach to combating the problem of short user queries. There are two approaches to QE, manual and automatic. Manual QE requires effort on the user's part. The amount of user effort to properly expand a query is considerable and is fraught with dangers. Failure to select the proper terms to be included in the expanded query can have a significant impact on the results obtained using the expanded query. Even if a good selection of expansion terms are used we then have the problem of properly weighting these terms so that they compliment the initial query and do not shift its focus away from its original goal. Over-expansion of the initial query can often lead to a shift in the focus of the query to something completely different which will result in degraded system performance in the users eyes.

An alternative approach to manual expansion is to automatically select terms to be included in the expanded query. The problem then becomes what to use as a source from which to get the additional terms. The obvious solution is to use the ranked list of documents generated as output from the running of the initial query. The idea of treating the top documents as being potentially relevant in the absence of any real relevance judgements is not a new one [Buckley *et al* 1994]. The top ranked documents in this list have a better than random chance of being relevant to the user's query and therefore become a good source of candidate terms for inclusion into the

query. This is not to say that this approach hasn't its flaws, if none of the top documents are relevant then the expansion is likely to have a very negative effect as the expanded query will emphasise the same mistakes that caused the poor initial retrieval. The net result of including QE in an IR system is to cause improvements for many queries, but deterioration for others. Research carried out by [Buckley *et al* 1994] has shown that an IR system's effectiveness improves linearly as the log of the number of terms added up to a point of diminishing improvements. The point can be made that how can so many terms be added when it is known that many of them are poor terms and have no connection with relevance. One contributing factor is simply that the good terms tend to co-occur non-randomly within the relevant documents (as opposed to the rest of the collection) and the poor terms tend to co-occur randomly. Massive query expansion establishes a background 'noise' similarity due to random poor term matches. The good documents escape the noise due to having several good terms co-occur within the document.

It must be noted that QE can modify the initial query in one of two ways. Firstly, new terms can be added to the query and secondly, existing terms can be re-weighted, shifting the focus from one part of the query to another. The approached we adopted in our system is to run the initial query, rank the results of the query and select the top *X* documents as the source for candidate expansion terms. An analysis of document size over the entire *TREC* collection illustrates that on average documents are too large to be treated as expansion units, i.e. there is too much information in a unit document. To combat this we used the statistical positional information stored in our inverted index structure to construct a range within each document within which all of the initial query terms occur, as illustrated in Figure 5.20.

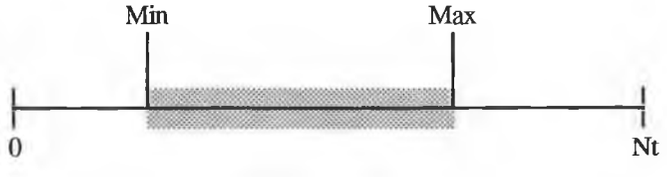


Figure 5.20 - Expansion unit restriction based on positional data.

This range of text within a document can be used as an expansion unit instead of using the whole document. In practically all situations the imposition of this range

restriction reduces the expansion unit size. Figure 5.20 shows an expansion unit (whole document or document passage, depending on whether the IR system has implemented passage level retrieval). The expansion unit is of length N_t (index terms). The first and last occurrences of all of the terms used in the query are computed. The minimum and maximum values are read and then used as the range boundary to further restrict the size of the expansion unit. This could result in only a paragraph from within a document being used as an expansion unit due to the fact that all of the initial query terms occur within the paragraph. It is therefore logical to use such a paragraph as a source of expansion terms.

If passage level retrieval is not in operation the expansion units can be highly variable in length therefore a normalising or weighting procedure must be applied to ensure equal treatment of terms from each document. To this end we developed a weighting scheme based on the expansion unit's initial query similarity score, its length and its component terms *IDF* score which automatically selects the 'best' expansion terms from the top X expansion units, 'best' meaning terms which occur relatively rarely within the corpus as a whole but occur frequently within the top X expansion units. Once the set of expansion terms have been generated they must be incorporated into the initial query to form the expanded query. New terms are added and the existing terms are re-weighted based on the expansion terms.

Once the expanded query has been created it is then processed in exactly the same manner as the original query, i.e. phrases are identified, stopwords are removed, the remaining words are stemmed. These stemmed words are searched for in the inverted file lexicon. If they occur then their respective posting lists are processed and partial query similarity weights are accumulated. The approach we took was to have a clean start for the expanded query and treat it as a completely separate entity to the original query. In practice this means resetting all of the original query term's non-zero accumulators to zero and resetting all threshold accumulators to zero as well.

Two additional thresholds were imposed on the query expansion process in an effort to ensure that the focus of the original query is not lost when it is expanded. These additional thresholds are based on the specificity of the original terms. When processing the original query a note is made of the maximum (most specific) and

minimum (least specific) IDF scores of the terms in the query. These minimum and maximum IDF values are then used in the query expansion process to limit the expansion terms included in the expanded query to those which fall between these minimum and maximum values. This expansion thresholding approach is derived from [Luhn 1958] (see Chapter 1) who states that terms which occur very frequently and terms which occur very infrequently are not the best document discriminators. Initial experiments carried out by us resulted in a large number of expansion terms being selected that were either very infrequently occurring or very frequently occurring. The very infrequently occurring (high *IDF* scores) tended to be highly specific phrases contained in one of the top X expansion units. The very frequently occurring terms tended to be commonly occurring terms that were not classed as stopwords due to the fact that they might be valid search terms on some occasions. Results showed that the infrequently occurring terms caused a focus shift from the original query due to a number of high IDF values being processed in the expanded query. Results also showed that the inclusion of the frequently occurring non-stopword terms swamped the accumulators with lots of low partial similarity scores which also caused a degradation in performance.

Rather than impose a global maximum and minimum IDF threshold we felt it would be better to base these thresholds on the original query itself. Obviously the more specific a query is the better. To this end our expansion procedure also tried to move the average query term IDF score towards the maximum IDF value and away from the minimum IDF value. This results in an expanded query with more of its terms having IDF sores close to but not greater than the original query's maximum IDF value.

5.4 Summary.

In this Chapter we described the techniques and approaches used during the development of our IR system. The IR system can be split up into two distinct components, the indexing engine and the retrieval engine. We described the components of and data structures used in the index creation process. The component parts and the data structures used during the retrieval process were then explained. In the next Chapter we define the notion of a query space and its implications for the

retrieval process along with detailing approaches to reducing the amount of the query space processed during retrieval. In the next Chapter we introduce the concept of Query Space Reduction and detail its requirements and implications for the retrieval process.

6. Query Space Reduction.

6.1 Introduction.

The issue of query response time is critical to our research. It is, in our opinion, very important that IR systems return the required information in an acceptable amount of time to the user and for us this is of equal importance as the effectiveness of an IR system. Traditional IR research has always concentrated on effectiveness and efficiency has been a poor relation. To this end we developed and implemented a number of Query Space modelling techniques which we believe will improve the efficiency of our experimental IR system. This Chapter informally describes these modelling techniques. Firstly, an abstract definition of a Query Space is given, secondly, two Query Space thresholding approaches and their effect on the Query Space are described, thirdly, our document accumulator restriction approach along with its impact on the Query Space is described and lastly a simulation of the operation of these thresholding approaches on the Query Space is presented.

6.2 Query Space Definition.

Within our test environment we define the Query Space (QS) to be the amount of data from the postings file which needs to be processed in order to satisfactorily respond to a users query. Figure 6.1 illustrates an abstract view of this data. The QS is composed of query terms and their corresponding posting lists. Query terms in the QS are those which occur both in the query text and the document collection, i.e. their inclusion in the QS will have some impact on the final ranking of documents returned to the user in response to the query. For visualisation purposes the QS is best laid out in two dimensional space with the Y axis representing the query terms and the X axis representing the posting lists of the query terms. Within the QS the query terms are sorted in order of posting list length. The query term with the smallest posting list length will be positioned at the top end of the Y axis and the query term with the longest posting list length will be positioned at the bottom end of the Y axis. This ordering is monotonic with the Inverse Document Frequency (*IDF*) score of each term. This *IDF* value is computed by the following formulae $IDF = \log(N / n)$ where N is

the number of documents in the entire document collection and n is the number of those documents the index term actually occurs in.

The posting lists themselves are ordered and there are three possible approaches to the ordering of posting lists each with their own advantages and disadvantage. These orderings are by:

- Increasing document identifier.
- Decreasing within document index term frequency.
- Decreasing within document index term density.

The ordering method employed has implications for both the indexing process and the retrieval process. Ordering by increasing document identifier facilitates easier insertions to the inverted file structure as new document postings are always appended at the end of the existing posting lists. For deletions, document postings scheduled for deletion are easily located within the posting list. However, this ordering approach eliminates the possibility of applying thresholding techniques to processing the posting list during the retrieval operation because if the posting list entries are sorted in order of increasing document identifier then the postings are effectively in random order for their respective index terms.

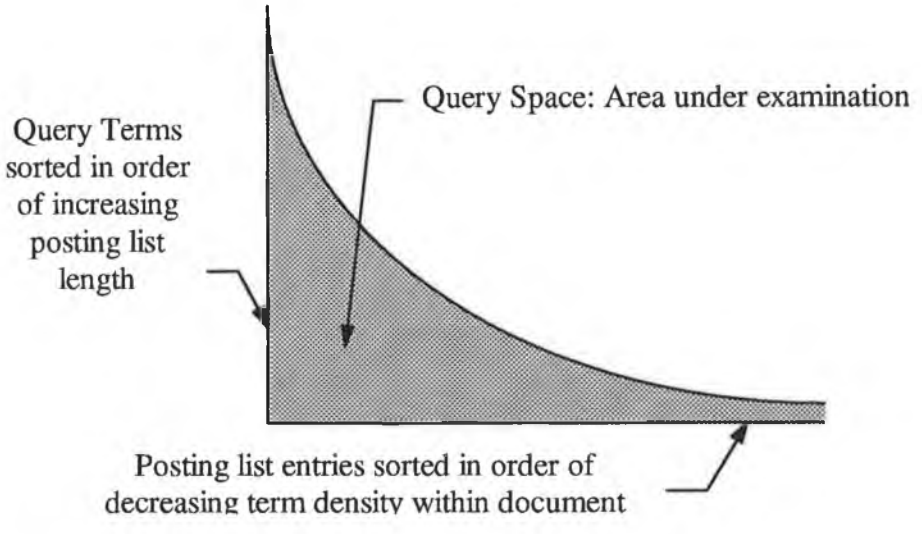


Figure 6.1 - Abstract View of Query Space.

This ordering approach results in degraded system performance during retrieval due to the necessity of having to read in and process the entire posting list in order to

extract the most valuable posting list entries with respect to the index term. Ordering by decreasing document term frequency and document term density share the same implications for the indexing and retrieval processes. This ordering approach makes updates to the inverted file structure more difficult because new document postings added to the index need to be inserted into the existing posting lists and not simply appended. The location of document postings scheduled for deletion also becomes more difficult as the entries in the posting list are not ordered in an easily accessible manner. To overcome this indexing maintenance problem an additional function would be required in order to carry out modifications to the inverted file structure. This additional function would firstly, read in the entire posting into memory, secondly, order it by increasing document identifier, thirdly, carry out the necessary additions and deletions, fourthly, re-sort the posting back to its original order and lastly write the updated posting list back to the inverted file.

The advantage of using the decreasing document index term frequency and document index term density ordering approaches lies in their ability to facilitate thresholding of the postings lists during the retrieval process. This thresholding procedure removes the necessity for the retrieval process to read in entire posting lists into memory for processing during retrieval thus greatly reducing I/O during retrieval.

As the primary focus of our research is the implementation of an efficient and effective IR search engine we felt that the retrieval advantages of the decreasing document index term frequency and document index term density ordering approaches far outweighed the indexing disadvantages. As a result of this we eliminated the use of the increasing document identifier ordering scheme and concentrated on the second and third posting list ordering approaches.

6.3 Query Term Thresholding.

As stated in Section 6.2 the QS is ordered vertically on increasing posting list length which maps directly to increasing *IDF* scores. This effectively means that query terms located at the top end of the Y axis in the QS are likely to be more discriminating because they occur in fewer documents. These terms while contributing to the retrieval process by their likely discrimination between relevant and non-relevant

documents also have the additional advantage of having short posting lists (due to the fact that they occur in relatively few documents). This means that processing these terms is both extremely beneficial in terms of effectiveness and in terms of efficiency.

At the other end of the Y axis we have query terms which occur more frequently within the corpus. Their *IDF* scores will be lower due to their relatively high occurrence frequency. These terms represent a challenge to the efficient and effective operation of the retrieval process by firstly contributing little in terms of document discriminating power to the retrieval process and secondly, by taking up the vast majority of the processing and I/O overheads of the retrieval process due to their relatively long posting lists.

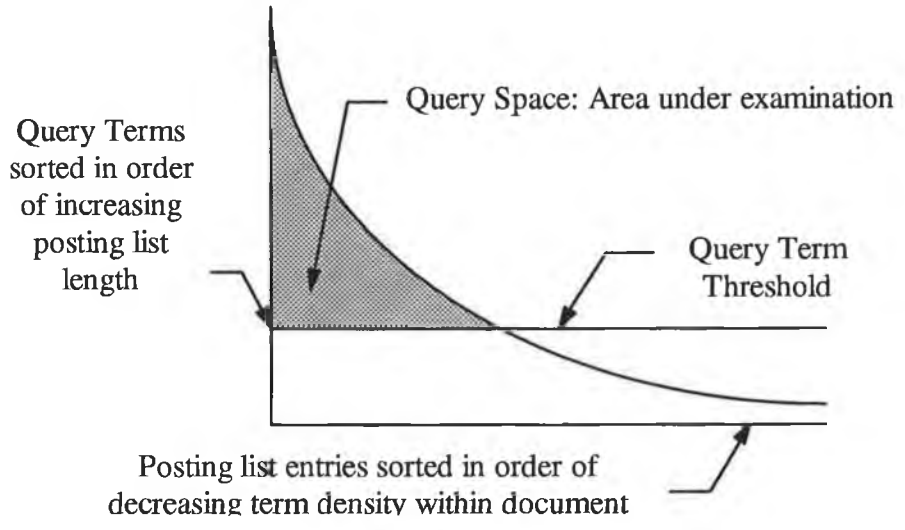


Figure 6.2 - Abstract View of Query Term Thresholding.

It therefore makes sense to attempt to reduce or eliminate the need to process these 'low value' query terms during the retrieval process. This is where the concept of Query Term Thresholding (*QTT*) comes in. *QTT* is a simple restrictive process in which query terms which have the longest postings entries above a certain threshold are not processed by the retrieval process. This has the effect of significantly reducing the processing and I/O cost of the retrieval process while also having a positive effect on the effectiveness of the retrieval process by eliminating 'noisy' postings from consideration during retrieval.

6.4 Posting List Thresholding.

The selection of our posting list ordering approaches allows us to implement our second form of QS thresholding, Posting List Thresholding (*PLT*). As the posting list entries are ordered in terms of decreasing value to their respective QS index term, posting entries at the end of a posting list will be of less value to the QS index term due to their low within-document frequency or within-document density and therefore the possibility exists of removing these 'low value' postings from consideration during the retrieval process. Posting entries at the end of posting lists of index terms with high *IDF* scores are more likely to be of use than posting entries at the end of posting lists of index terms with low *IDF* scores. This means that more of the discriminating posting lists (those with high IDF scores) entries and less of the non-discriminating posting list entries should be processed. This results in a variable thresholding approach in which the *PLT* value is initially set to a high percentage of postings and is gradually lowered as each QS index term is processed. Figure 6.3 illustrates this thresholding process in action. This thresholding approach has the effect of eliminating most of the 'low value' posting entries from the 'low value' QS index term posting lists.

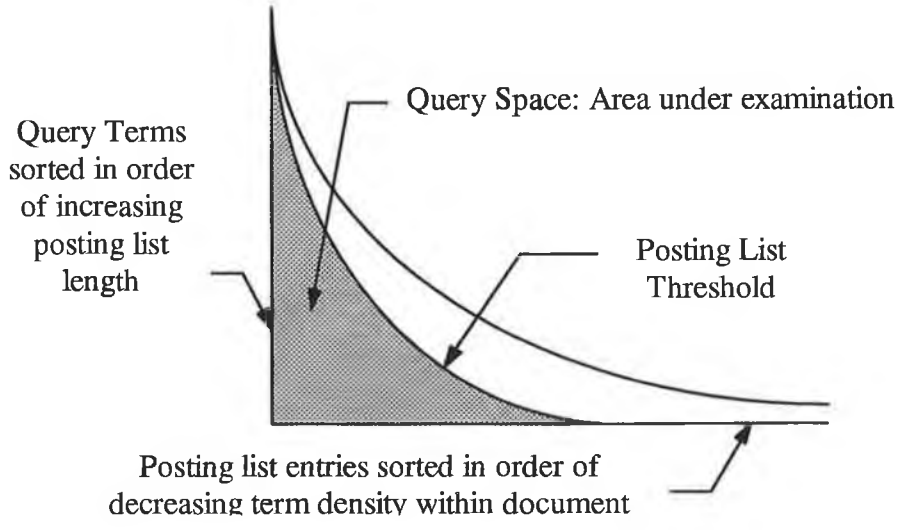


Figure 6.3 - Abstract View of Posting List Thresholding.

During the course of our research we developed and implemented two variations of the *PLT* procedure. The first *PLT* procedure is controlled by three values, firstly, the Starting Thresholding Value (*STV*), secondly, the Ending Thresholding Value (*ETV*)

and lastly, the Number of Query Terms (*NQT*) being processed. The *PLT* value starts at *STV*, and ends at *ETV* by decreasing in steps of $(STV - ETV)/NQT$. The second, modified *PLT* (*MPLT*) procedure holds the *MPLT* value at the *STV* value until $NQT/2$ QS index terms have been processed and it then decreases the *MPLT* value by $(STV - ETV)/(NQT/2)$. This has the effect of processing even more posting list entries of the most discriminating QS index terms.

6.5 Query Term and Posting List Thresholding.

The QS thresholding techniques detailed in Sections 6.3 and 6.4 can be combined and can operate in conjunction with each other on the same QS. When these two thresholding approaches are combined, the NQT value used by the *PLT* and *MPLT* approaches is that defined by the *QTT* approach. The combination of *QTT* and *PLT* or *MPLT* results in significantly lower processing and I/O overheads than processing the full inverted file entries during retrieval as will be shown in Chapter 7.

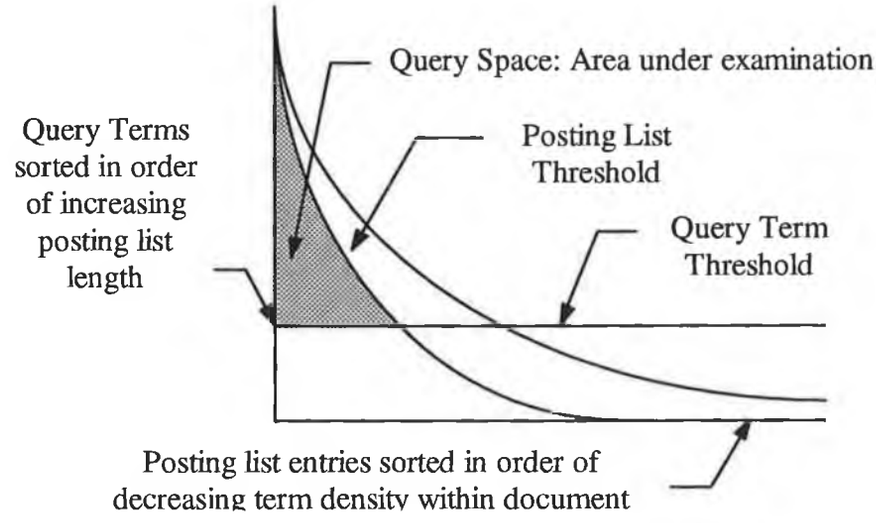


Figure 6.4 - Abstract View of Combined Thresholding Approach.

6.6 Document Accumulator Thresholding.

A document accumulator is a register used to hold partially computed document scores during processing of query terms. It has been shown by [Moffat & Zobel 1994] that even a six term query on average activates around 50% of the documents accumulators associated with a collection. If no restriction is imposed on the allowed

activation of document accumulators then a large number of accumulators must be sorted in order to extract the top N_r scored documents. Furthermore, the memory requirements for storing one accumulator per document (with 4 Bytes per Accumulator) for a collection of *TREC* proportions (~1.3 million documents) could be around 50 Mbytes in total. Therefore some method for limiting the number of accumulators activated during this phase of retrieval is very important. To this end we developed the following algorithm:

```
START
{
    while( end-of-query( ) == FALSE )
    {
        query-term = get-next-query-term( );
        posting-list = access-posting-list( query-term );
        while( end-of-posting-list( posting-list ) == FALSE )
        {
            extract-posting( posting-list );
            if( active-accumulators < max-active )
            {
                qds = calc-query-doc-sim( posting, query-term );
                add-new-accumulator( qds, posting );
            }
            else if( accumulator-active( posting ) == TRUE )
            {
                qds = calc-query-doc-sim( posting, query-term );
                accumulate-accumulator( qds, posting );
            }
        }
    }
}
END
```

Figure 6.5 - Restrictive Processing of Posting List Entries.

This result of this thresholding approach is that an upper limit is placed on the number of document accumulators allowed to activate. New accumulators are created for all documents which achieve a non-zero query document similarity score until the maximum limit of accumulators is reached. Once reached only already activated accumulators are allowed accumulate more partial query document similarity scores. This has the effect of controlling the number of accumulators activated hence reducing the amount of data which must be sorted in order to produce a ranked list of results.

6.7 Summary.

In this Chapter we introduced the concept of a 'Query Space' being the body of data that must be processed in order for an IR system to satisfactorily respond to a query. We then outlined a number of query space thresholding techniques that may be

employed by an IR system to reduce the amount of the query space that must be processed during retrieval. We also described the purpose of document accumulators and explained their role in the retrieval process along with introducing thresholding techniques which restrict the number of document accumulators active during the retrieval process. In the next Chapter we go into more detail about these Query Space thresholding approaches by reporting on experiments we carried out to evaluate these thresholding techniques in a realistic environment. In the next Chapter we describe in detail the experiments we carried our within our test environment in order to assess and evaluate the impact our Query Space thresholding approaches have on retrieval performance.

7. Experimental Runs.

7.1 Introduction.

This Chapter will detail all of the experiments carried out during the course of this research. All of our experiments were carried out using the *TREC* text collection. This collection has been expanded over the past three years to an overall size of 2.2 Gigabytes. The entire collection is split up into three overlapping sub-collections namely, the *TREC*-3 collection, the *TREC-4* collection and the TREC-5 collection. Each sub-collection has associated with it its own set of queries and corresponding relevance judgements, thus our experiments are run on the *TREC-3*, *TREC-4* and *TREC-5* sub-collections with different query sets and corresponding relevance assessments. While there is a certain amount of overlap between the sub-collections in terms of the document text each sub-collection's characteristics are significantly different from each other so as to provide a range of testing environments on which to carry out our experiments.

7.2 Purpose of Experiments.

The central purpose of the following set of experiments is to determine whether or not the thresholding approaches detailed in Chapter 6 coupled with the modified index structure detailed in Chapter 5 are of benefit in maintaining retrieval effectiveness while improving retrieval efficiency. This question is the essence of the thesis and the results presented herein are analysed later. Within our experimental environment we measure effectiveness through the use of Precision Recall (PR) graphs. Precision is the ratio of the number of relevant documents retrieved to the total number of documents retrieved and recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents (both retrieved and not retrieved). From the PR graphs generated from each experimental run we were most concerned with the number of relevant documents returned and the average precision because these values provide us with the clearest indication of how the system is performing in terms of effectiveness. Efficiency within our experimental environment is measured in terms of time taken, CPU usage and memory usage.

7.3 Hardware Resources Used.

All of the following experiments were carried out on a SUN SparcStation 5 with 64 Mbytes of RAM and 6 Gbytes of local disk space, running at a clock speed of 110 MHz on a microSPARC-II and using the SOLARIS operating system. While this machine was connected to the local Ethernet network we had exclusive access to the machine's resources for the duration of these experiments. All timing measurements presented in this Chapter were obtained using the UNIX 'time' command.

7.4 TREC-3 Experiments.

Within the bounds of this *TREC-3* collection we carried out a number of experiments on all of the thresholding approaches we developed during the course of our research. Each thresholding approach was tested individually in order to determine its impact (in isolation) on efficiency and effectiveness of the IR engine. The thresholding approaches were then combined in order to determine their collective impact on the system.

7.4.1 Accumulator Restrictions in *TREC-3*.

The first set of experiments on the *TREC-3* sub-collection were carried out to determine the effect of the imposition of an upper limit on the number of accumulators allowed to be activated in response to a query. For these experiments the query set (queries 151 to 200) and the relevance judgements associated with the *TREC-3* sub-collection were used. The top 1000 documents for each query were passed to the *TREC* evaluation program which takes the candidate set of relevant documents and generates averaged Precision-Recall figures with respect to the sub-collection's set of known relevance judgements.

In order to investigate the effect of the accumulator restriction in isolation all other parameters passed to the IR engine such as QTT and PLT thresholds were frozen and only the maximum number of accumulators allowed activate per query was varied from 5,000 to 120,000. We monitored the number of relevant documents returned in the top 1000 in the ranking (see Figure 7.1) and the Average Precision (see Figure 7.2).

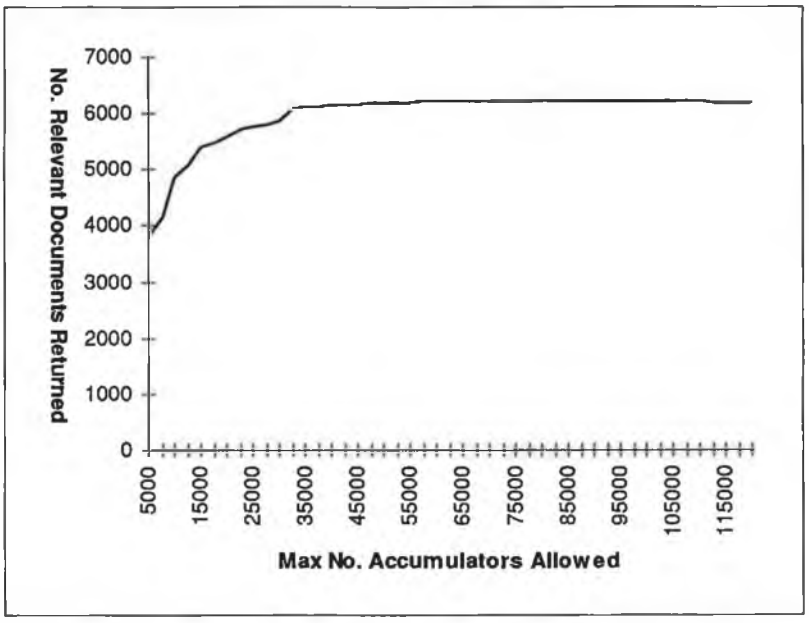


Figure 7.1 - *TREC-3* Accumulators Used Vs Relevant Documents.

The most striking result of this set of experiments is the fact that for the most part retrieval effectiveness is unaffected by the imposition of the accumulator restriction except at a low number of accumulators. The number of relevant documents returned and the average precision (in response to a set of 50 queries) is not impaired by the imposition of the accumulator restriction until that restriction becomes very severe (in this instance less than 30,000 accumulators).

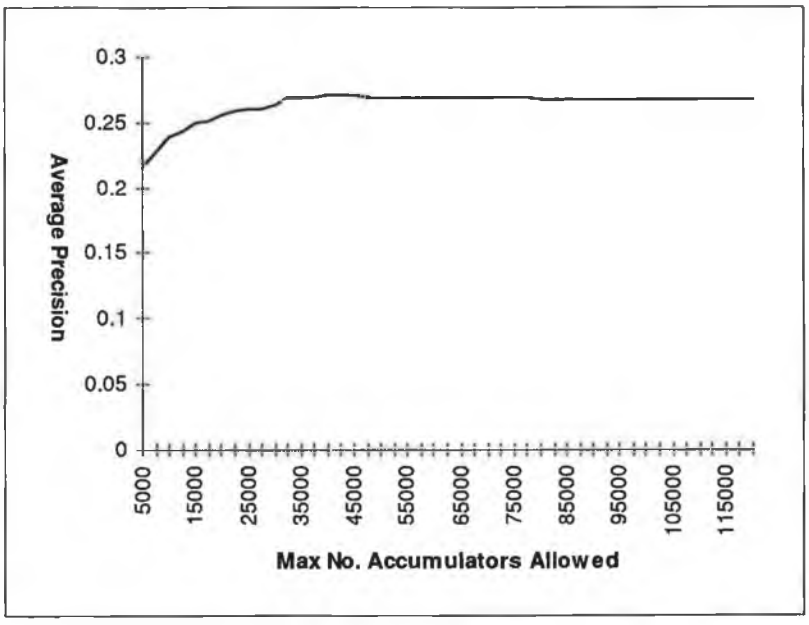


Figure 7.2 - *TREC-3* Accumulators Used Vs Average Precision.

The Y axis in Figure 7.1 represents the number of relevant documents returned in the top 1000 rank positions over 50 queries with the total number of relevant documents for the *TREC-3* collection being 9,805. We also monitored the time taken, CPU and memory usage for each query batch, which corresponds to a set of 50 queries.

Max Accumulators	5,000	35,000	50,000	70,000	90,000	117,500
Time (Per Batch)	894	1,422	1,515	1,620	1,703	1,884
Time (Per Query)	17.88	28.44	30.3	32.4	34.06	37.68

Table 7.1 - *TREC-3* Accumulator Timings (in Seconds).

The results, as illustrated in Figure 7.3 and Table 7.1 show that the time taken to process each query batch increases as the maximum number of accumulators is increased. When the timing values are brought down to a per query basis we see a 52.54% reduction in the time to process a query using 5,000 accumulators as opposed to 120,000 accumulators. This improvement does come with a reduction in effectiveness in terms of relevant documents retrieved (-38.8%) and in terms of average precision (-19.23%). However these figures apply to the most restrictive accumulator value. If this restriction is relaxed to allow 30,000 accumulators activate per query we have a totally different situation.

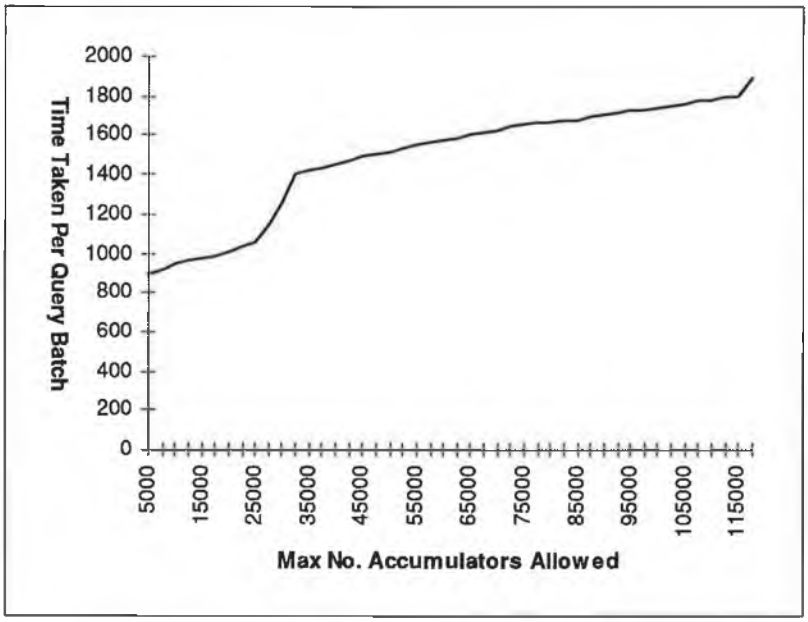


Figure 7.3 - *TREC-3* Accumulators Used Vs Time Taken (in Seconds)

We still achieve a large drop in the average time taken to process a query (-24.52%) but the corresponding drop in effectiveness in term of number of relevant documents returned (-1.43%) is minimal and average precision is actually slightly better (+0.74%). In order to illustrate the benefits involved in incorporating accumulator restrictions into the retrieval process we normalised the effectiveness graph and the efficiency graph into the range [0...1] and combined the result into Figure 7.4. It can be clearly seen from the graphs in Figure 7.4 that a large improvement in efficiency is possible without any adverse effect in effectiveness both in terms of the number of relevant documents returned and in terms of the average precision. The result of this experiment allows us to reduce the number of accumulators allowed to be activated to around 35,000 per query with no noticeable impact on effectiveness. If speed of response is the most critical factor then the maximum accumulator value could be reduced even further but this would result in the degradation of system effectiveness.

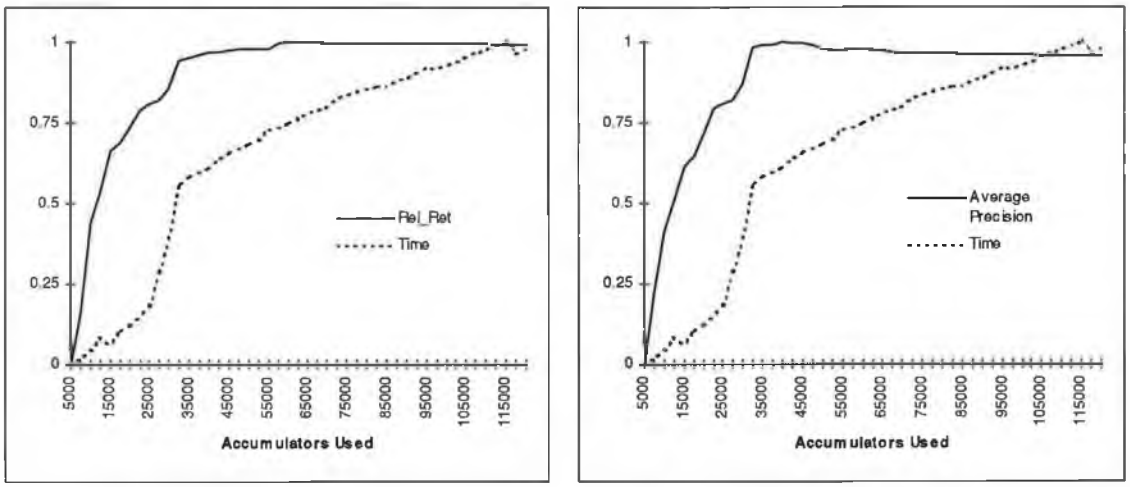


Figure 7.4 - *TREC-3* Accumulator Efficiency Vs Effectiveness.

7.4.2 Query Term Thresholding in *TREC-3*.

The next set of experiments we carried out were to investigate the effect of imposing *QTT* on the retrieval process. *QTT* as already detailed in Chapter 6 controls the number of terms within a query that are actually included in the retrieval process for that query. As with the accumulator experiments we froze all other parameters to the IR engine and varied only the *QTT* value from 1% to 100%. At 1% only query

terms that occur in less than 1% of the documents in the collection are processed, at 100% all query terms are processed by the IR engine.

For this set of experiments we set the maximum document accumulator value to the maximum value of the previous set of experiments (120,000). Figure 7.5 details the effect of the *QTT* percentage on the number of relevant documents returned in the top 1000 in response to a query batch. It can be seen that the number of relevant documents returned remains largely unaffected by the *QTT* threshold value until that value becomes very restrictive. It is only when the *QTT* value is set below 10% do we notice a drop in effectiveness in term of the number of relevant documents returned.

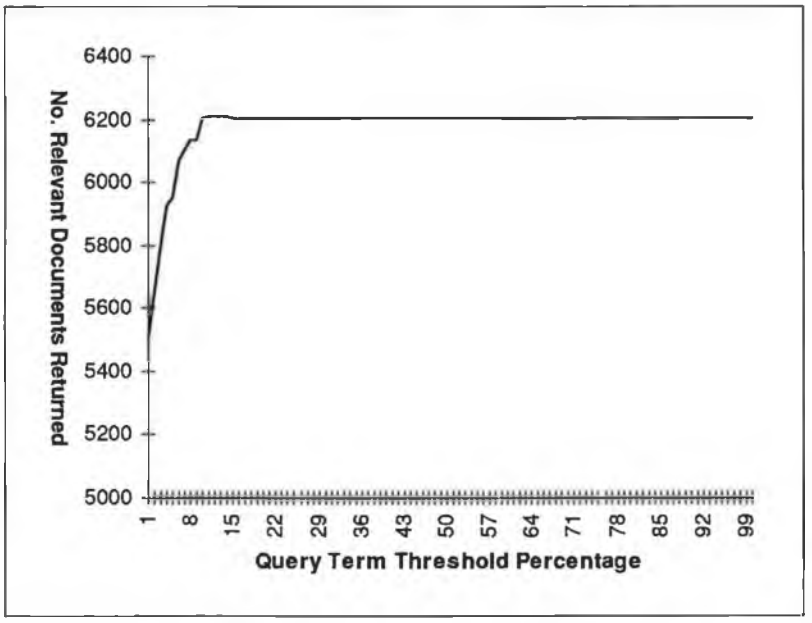


Figure 7.5 - *TREC-3 QTT* Percentage Vs Relevant Documents.

The same observation applies to the average precision values, i.e. like the number of relevant documents returned per query set they do not degrade significantly until the *QTT* setting is lower than 5%. It is also significant to note that the average precision peaks when the *QTT* percentage is between 5% and 20% and degrades somewhat when the *QTT* percentage is increased above 20% at which stage it flattens out. A reason for this is that as the *QTT* percentage is increased more and more general (but non stopword) terms are included in the retrieval process.

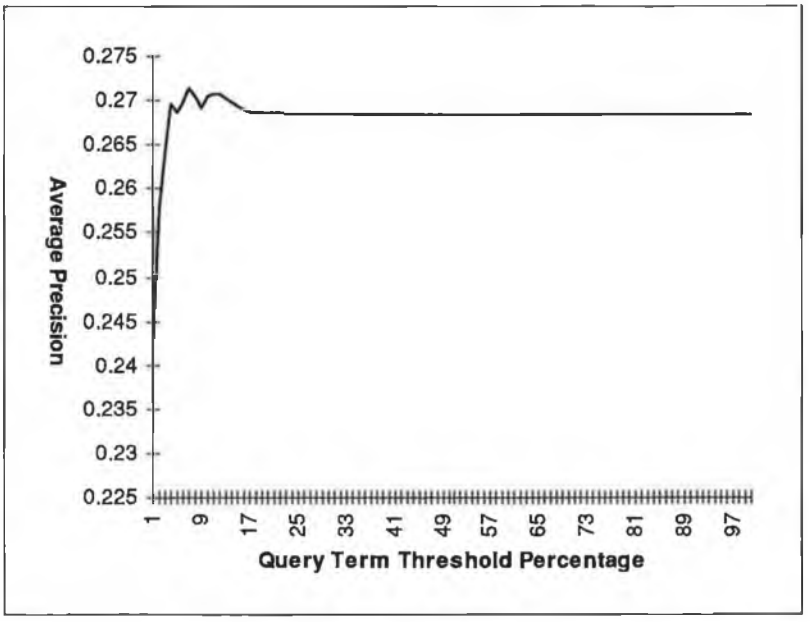


Figure 7.6 - *TREC-3 QTT* Percentage Vs Average Precision.

These general terms contribute little in terms of partial similarity scores however because of the sheer volume of occurrences of these general terms within the collection they have a tendency to swamp the retrieval process and degrade overall performance in terms of efficiency (the need to process them) and effectiveness (loss of significant partial similarity scores by the accumulation effect of so many small partial similarity scores).

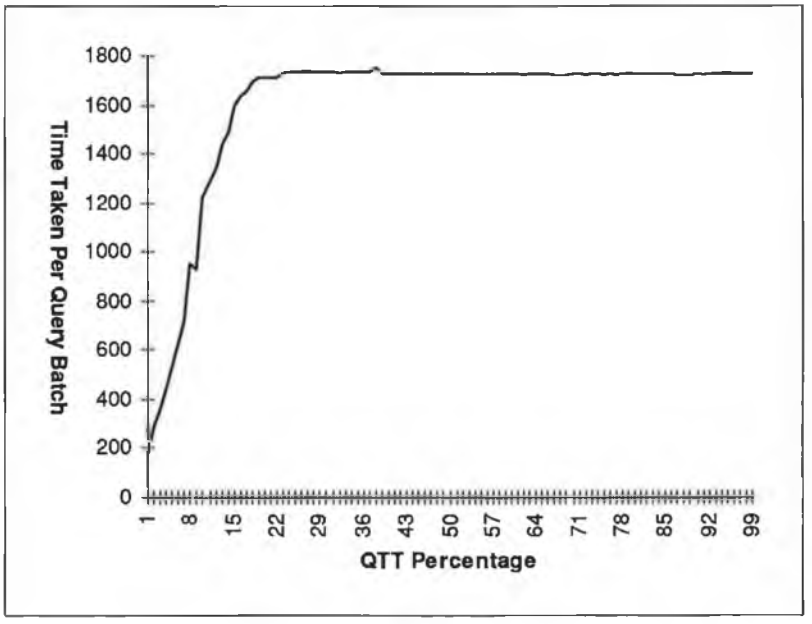


Figure 7.7 - *TREC-3 QTT* Percentage Vs Time Taken (in Seconds)

Figure 7.7 details the effect the *QTT* percentage used has on the time taken to process each query batch. It is very interesting to compare Figure 7.7 with Figure 7.5 and Figure 7.6. Figure 7.7 illustrates clearly that the time taken (in seconds) to process each query batch start decreasing significantly when the *QTT* percentage drops below 25%. However there is no corresponding drop off in performance in terms of number of relevant documents retrieved until the *QTT* percentages drops below 10% and in terms of average precision until the *QTT* percentage drops below 5%. This means that when the *QTT* percentage is dropping from 25% to between 5% and 10% we have significant improvements in terms of efficiency with no corresponding drop in effectiveness. From Figure 7.7 is can be seen that the graph flattens out after the *QTT* percentage reaches 25%; this means that for this particular query set the vast majority of query terms occur in less than 25% of the documents so once the *QTT* percentage reaches 25% or above the IR engine is not eliminating any of the query terms from the retrieval procedure hence there is no reduction in the time taken to process the query batch. All of the above comparisons between efficiency and effectiveness when using *QTT* are summarised in Figure 7.8. The results illustrated in Figure 7.5, Figure 7.6 and Figure 7.7 are normalised into the common range [0...1] in order that they may be overlaid and presented in Figure 7.8. Figure 7.8 clearly shows that efficiency improvements can be achieved without any impact on effectiveness both in terms of average precision and the number of relevant documents returned.

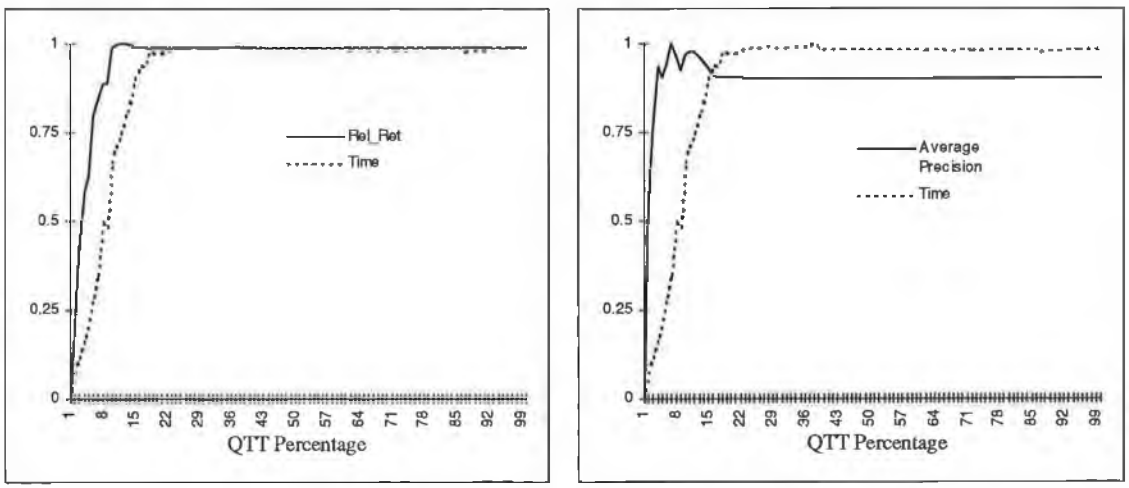
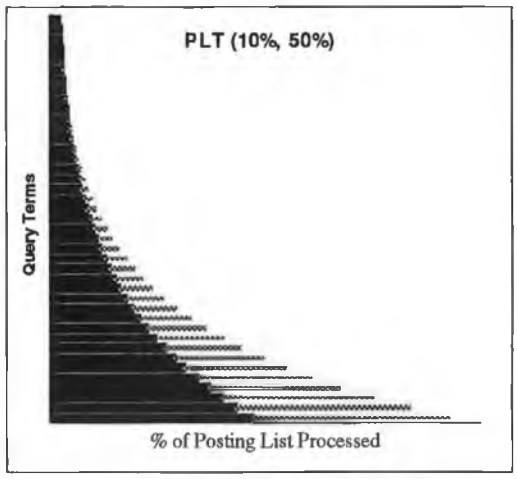


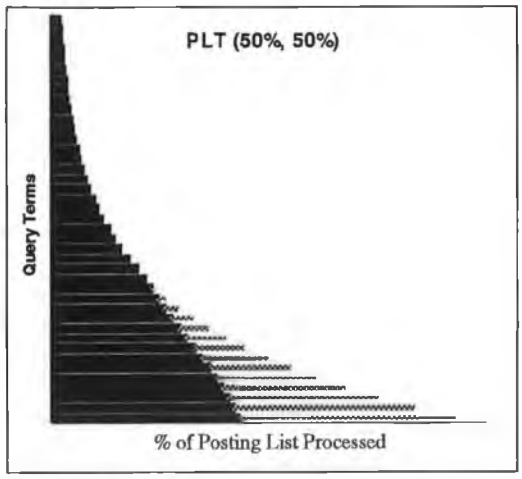
Figure 7.8 - *TREC-3 QTT* Efficiency Vs Effectiveness.

7.4.3 Posting List Thresholding in *TREC-3* .

The next set of experiments we carried out was to investigate the effect of imposing *PLT* on the retrieval process. *PLT* as already detailed in Chapter 6 controls the amount of each posting list that is processed during the retrieval process. As with the previous experiments we froze other parameters to our IR engine such as document accumulator restriction and only varied the *PLT* threshold values. The *PLT* thresholding process is controlled by two threshold values 1) the starting threshold value and 2) the ending threshold value. The starting threshold value determines when *PLT* takes place the ending threshold value determines how much *PLT* take place for the last of the query terms that are processed. In these experiments the starting and ending threshold values both ranged from 2% to 99% with the starting thresholding value ascending and the ending threshold value descending as follows: (2%:99%,... 12%:89%, ... 99%:2%).

As with the previous experiments we monitored the effect of the thresholding (*PLT*) on the number of relevant documents returned in the top 1000 and the average precision value. In addition, we also monitored the efficiency of the retrieval process in terms of the time taken (in seconds) to process each batch of 50 queries. At this point it would be useful to abstractly illustrate the effect of the *PLT* approach has on the QS as follows:





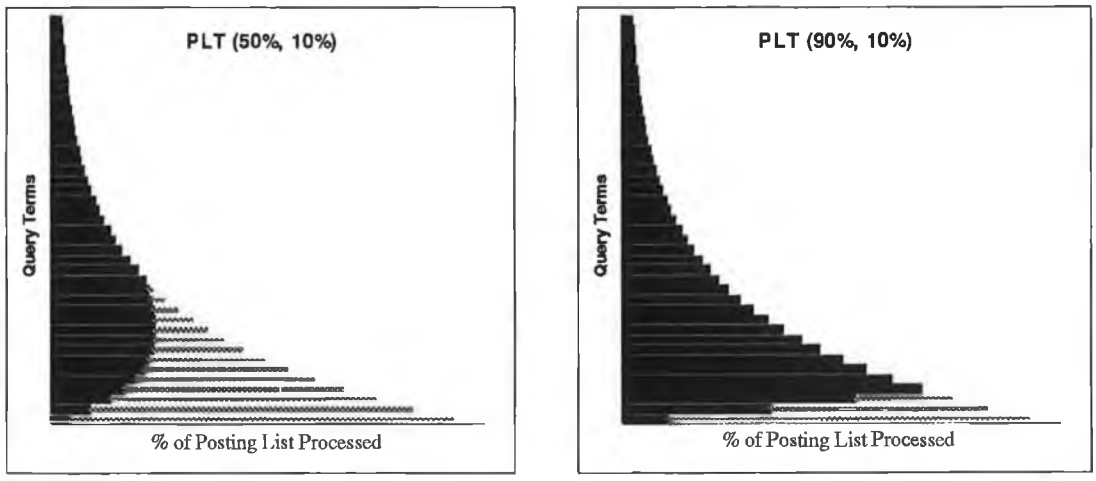


Figure 7.9 - Abstract View of various *PLT* settings.

Figure 7.9 illustrates four examples of *PLT* in operation. In all examples the y-axis represents the set of query terms in the current query sorted from top to bottom by decreasing IDF value and the x-axis represents the posting list entries sorted from left to right in order to decreasing within document density. The curved *PLT* boundary is due to the *PLT* starting and ending values being percentages of the actual value and therefore relative rather than absolute percentages. The top left abstract QS in Figure 7.9 shows the starting threshold value set to 0% and the ending threshold value set to 50%. This results in thresholding being applied to all of the posting lists in the QS. The ending threshold value comes into operation immediately. The amount of posting entries in each posting list discarded during processing is determined by the position of the term in the term rankings, by the length of the posting list and the starting and ending threshold values. The percentage discarded for each posting list is linearly reduced from 100% of the starting posting list (determined by the *PLT* starting percentage) to the ending percentage which is applied to the last posting list in the current QS. The top right abstract QS in Figure 7.9 shows the *PLT* settings of 50% and 50% meaning that the first half of the posting lists in the QS are not restricted in any manner and the second half of the posting lists are restricted linearly from 100% to an ending percentage of 50% for the longest posting list (in this instance the last). The bottom left abstract QS in Figure 7.9 illustrates a more severe thresholding setting of 50% and 10% for the starting and ending *PLT* values while the bottom right abstract QS in Figure 7.9 show us a relaxed *PLT* setting with 90% of the posting lists not restricted in any manner.

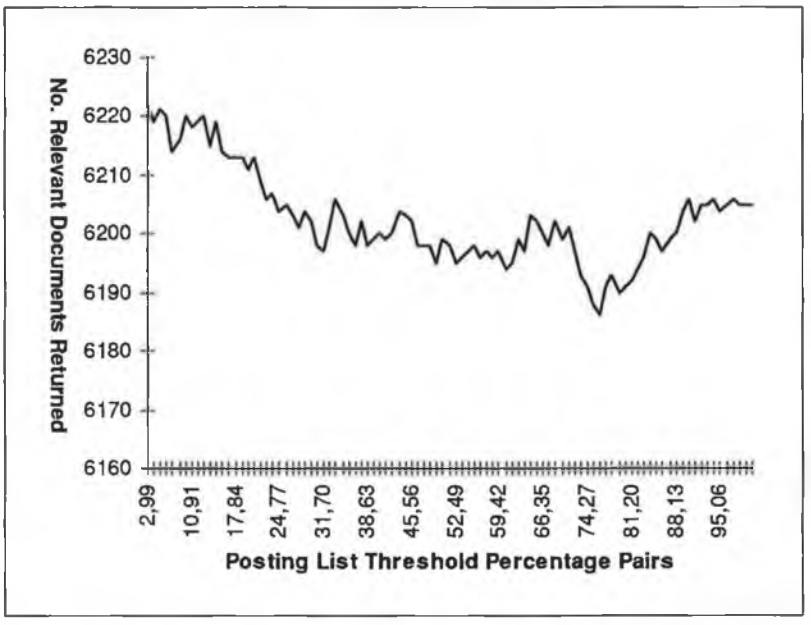


Figure 7.10 - *TREC-3* Posting List Thresholding Vs Relevant Documents.

Figure 7.10 illustrates the effect of various *PLT* settings on the number of relevant documents returned per query set. It can be seen that as the *PLT* threshold value approach the 50%,50% setting the number of relevant documents returned also approaches its minimum value after which the number of relevant document returned does rise again but not back to its original starting value. However it must be noted that the number of relevant documents returned does not drop significantly until after the *PLT* values reach a 15%,85% setting. This provides the possibility of achieving efficiency improvements by *PLT* without compromising effectiveness. The amount of degradation as measured by the number of relevant documents returned in the top 1000 is very slight with no PLT we retrieve 6221 relevant documents, at the worst PLT performance settings this drops only 0.57%, to 6185 relevant documents.

A similar result can be found for the average precision values when measured against *PLT* settings (see Figure 7.11). These follow a similar trend to the number of relevant documents returned, i.e. the average precision is relatively unaffected until the 15%,85% *PLT* settings are reached after which there is a steady drop off until the 50%,50% *PLT* setting followed by a small rise in the average precision value as the *PLT* settings approach the 99%,2% values. The amount of degradation in terms of average precision is also very slight, with no PLT we get an average precision of 0.2745, at the worst PLT performance settings this drops only 2.36% to .2680.

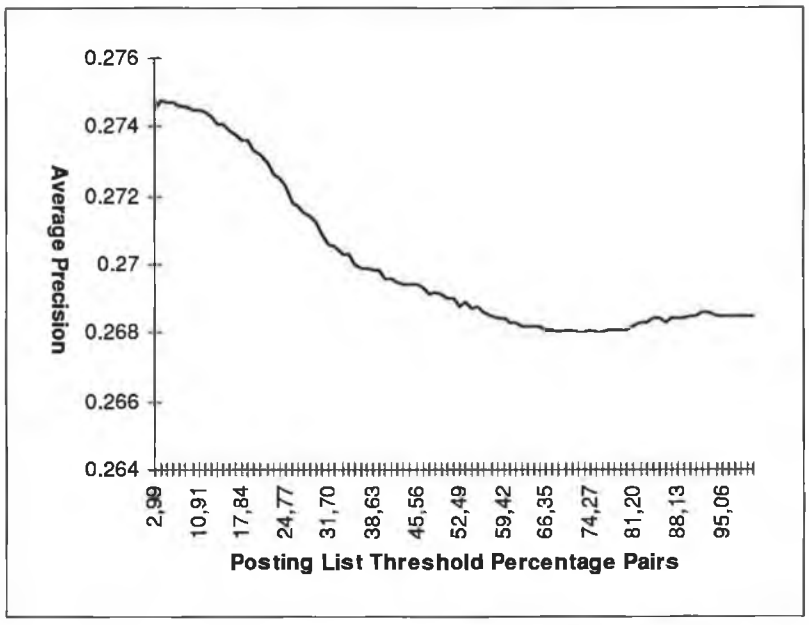


Figure 7.11 - *TREC-3* Posting List Thresholding Vs Average Precision.

In terms of reducing the amount of posting list entries processed, a *PLT* setting of 50%, 50% for the starting and ending threshold values respectively should and in fact does yield the best performance in terms of the time taken (in seconds) to process each query batch. As expected, the graph of the efficiency criterion is relatively symmetric i.e. *PLT* settings of 2%, 99% and 99%, 2% both take the roughly same time to complete.

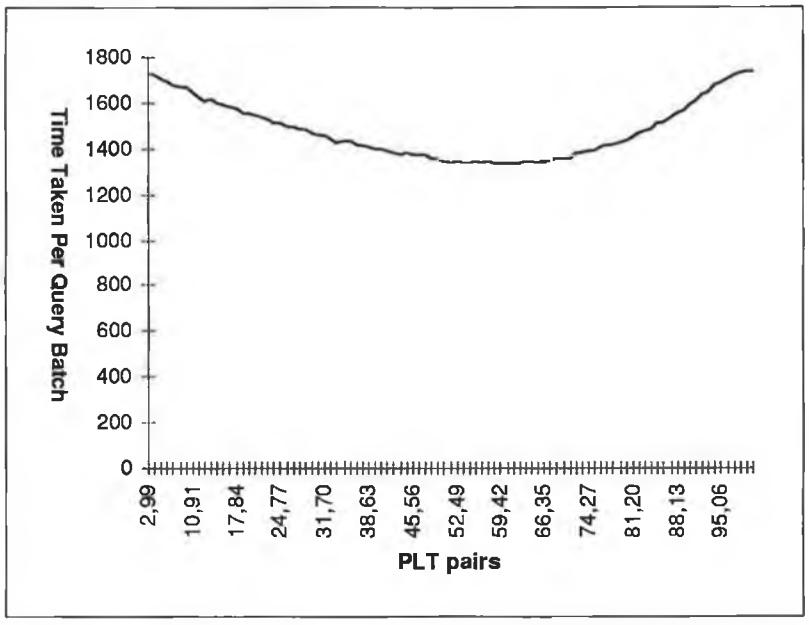


Figure 7.12 - *TREC-3 PLT* Percentages Vs Time Taken.

It can be seen from Figure 7.12 that the minimum point is not exactly at the midpoint, and actually occurs at the 56%, 45% setting. This can be explained if one considers the overall shape of the QS (narrow at the top and wide at the bottom). More savings in efficiency can be achieved by thresholding more of the lower part of the QS instead of the upper half. Once again in order to clearly view the impact of *PLT* on efficiency and effectiveness we normalised the results in Figure 7.10, Figure 7.11 and Figure 7.12 into the range [0...1] and overlaid them in Figure 7.13.

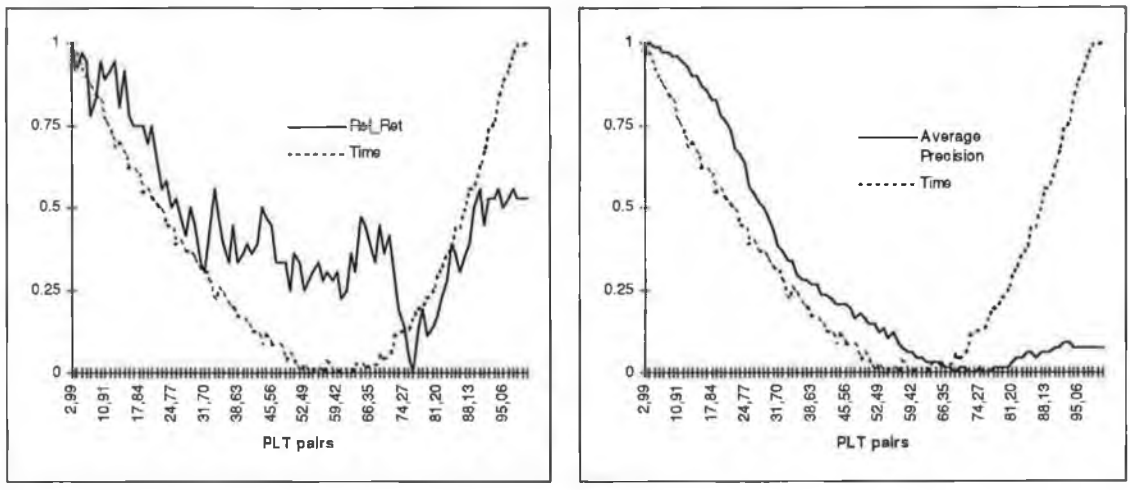


Figure 7.13 - *TREC-3 PLT* Efficiency Vs Effectiveness.

What can be seen from Figure 7.13 is that the potential for improving efficiency without compromising effectiveness exists with careful selection of the *PLT* settings. Any *PLT* setting up to 15%,85% will achieve improvements in efficiency without significantly degrading effectiveness.

7.4.4 Thresholding Combinations in *TREC-3*.

So far in our *TREC-3* experiments we have evaluated the impact of the proposed thresholding approaches (Accumulator Restriction, *QTT* and *PLT*) in isolation. The next logical step is to combine the above thresholding approaches in some fashion and evaluate their collective impact on retrieval performance both in terms of effectiveness and efficiency. To this end we have carried out a large number of experiments with the various thresholding approaches combined in many different manners. When combining the different thresholding approach in the one QS care must be taken that the individual thresholding approaches do not interfere with each others operation.

The Accumulator Restriction thresholding approach can easily operate without interfering with the operation of *QTT* and *PLT* due to its restriction criterion being based on size of the set of active accumulators. However care must be exercised in the combination of *QTT* and *PLT* because they both restrict the QS based on its shape. This resulted in the following algorithm being used to vary the necessary thresholding parameter settings over a range.

```
START
    for( mda = 30000 to 55000 step 5000 )
    [
        for( qtt = 3 to 21 step 3 )
        [
            for( plts = 10 to 70 step 10 )
            [
                for( plte = 85 to 95 step 5 )
                [
                    call_search( mda, qtt, plts, plte )
                ]
            ]
        ]
    ]
END
```

Figure 7.14 - Threshold parameter generation procedure.

The settings detailed in Figure 7.14 resulted in 882 unique parameter combinations, with each parameter combinations being applied to the set of 50 *TREC-3* queries.

	QSR
Retrieved:	50000
Relevant:	9805
Rel_ret:	**6221**
P. at 0.0	0.7174
P. at 0.1	0.5293
P. at 0.2	0.4508
P. at 0.3	0.3771
P. at 0.4	0.3222
P. at 0.5	0.2710
P. at 0.6	0.2261
P. at 0.7	0.1630
P. at 0.8	0.1058
P. at 0.9	0.0424
P. at 1.0	0.0003
Av. P	**0.2747**

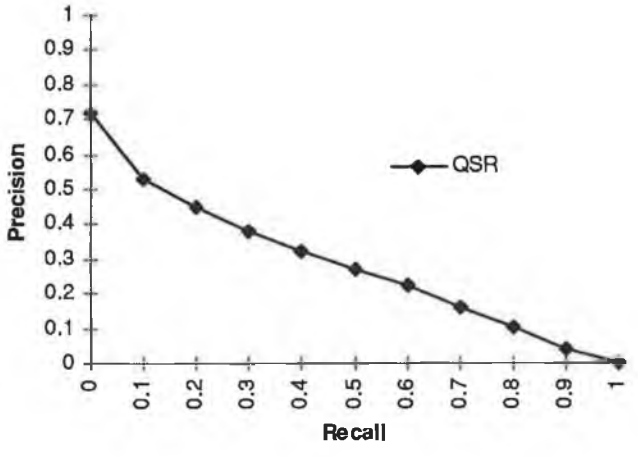


Figure 7.15 - Optimal Effectiveness Performance for *TREC-3* collection.

The output of each run of the IR engine was stored and passed to the *TREC* evaluation software which produced Precision-Recall figures for each run.

Figure 7.15 details the performance of the best parameter settings. These optimal settings are presented in Figure 7.16:

Parameter	Optimal Value
Query Term Thresholding:	6%
Posting List Thresholding:	*(Start)* 10%, *(End)* 90%
Maximum Document Accumulators:	50,000
Weighting Scheme:	log(tf) * (IDF^2) * qf
Document Normalisation Scheme:	No Normalisation
Document Fragmentation:	On *(Page Size 200 keywords)*

Figure 7.16 - Optimal System Parameter Settings (*TREC-3*).

7.5 TREC-4 Experiments.

In order to show that the results obtained using the above thresholding approaches individually and combined were not specific to the *TREC-3* sub-collection, we carried out the same set of experiments using the *TREC-4* sub-collection. The results are consistent with those from the *TREC-3* sub-collection. There were however slight differences in the results due to the radically different nature of the queries associated with the *TREC-4* sub-collection as detailed in Chapter 4.

7.5.1 Accumulator Restrictions in *TREC-4*.

The *TREC-4* queries were much shorter in length than the *TREC-3* queries (typically only one sentence). This means that the total number of accumulators activated by the *TREC-4* queries is less than the total number of accumulators activated by the *TREC-3* queries. This has the effect of allowing more thresholding of the accumulators without a corresponding drop in retrieval effectiveness in terms of the number of relevant documents returned and of average precision.

It can be seen from Figure 7.17 and Figure 7.18 that effectiveness does not significantly degrade until the maximum number of accumulators allowed to be active in response to a query is restricted to below 25,000 as opposed to 35,000 with the *TREC-3* sub-collection.

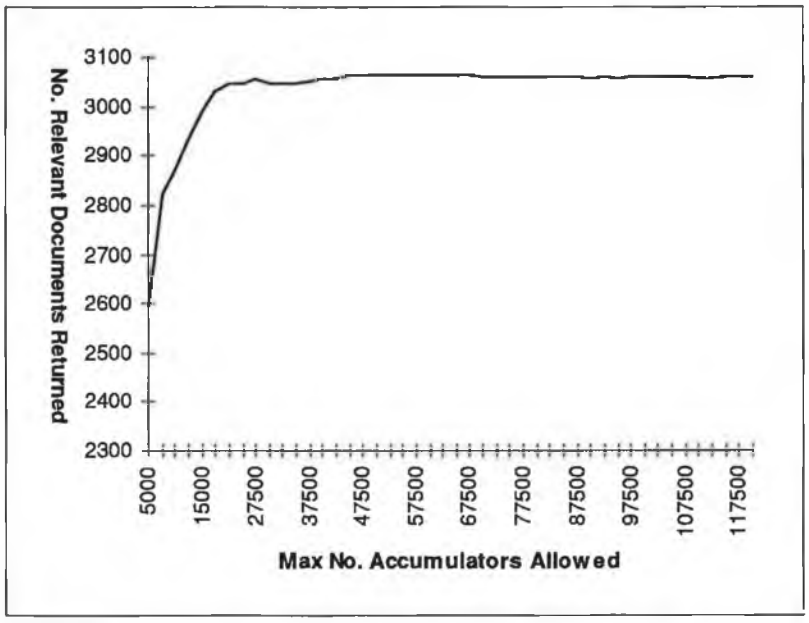


Figure 7.17 - *TREC-4* Accumulators Used Vs Relevant Documents Returned.

This would suggest that the limit on the maximum number of accumulators allowed activate can be linked to the number of query terms in the queries being processed.

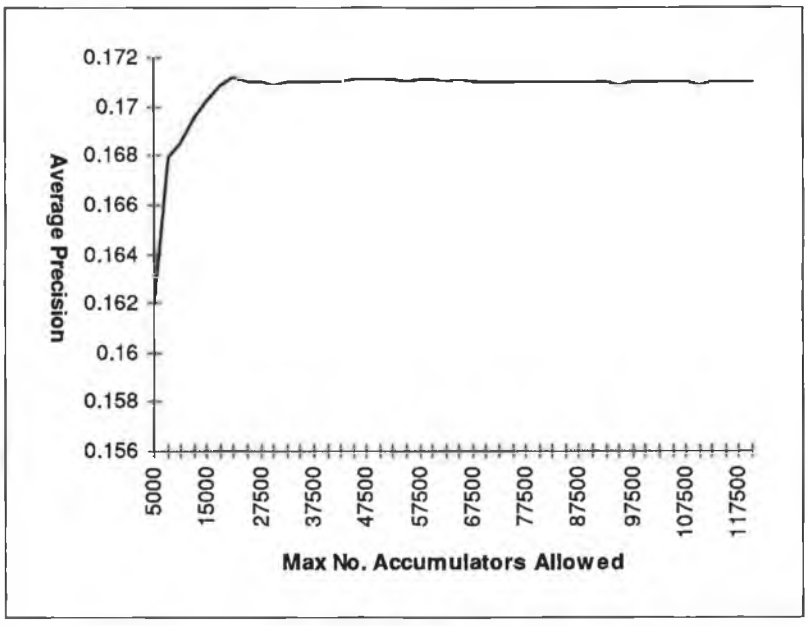


Figure 7.18 - *TREC-4* Accumulators Used Vs Average Precision.

Improvements in efficiency in terms of the time taken to process a query batch were similar to that found in the *TREC-3* collection. By comparing Table 7.2 and Table 7.1 it can be seen that it takes much shorter time to process the *TREC-4* query batches

than the *TREC-3* query batches anyway. This is due simply to the relative shortness of the *TREC-4* queries when compared to the *TREC-3* queries.

Max Accumulators	5,000	22,500	50,000	70,000	90,000	117,500
Time (Per Batch)	185	249	358	399	430	484
Time (Per Query)	3.70	4.98	7.16	7.98	8.6	9.68

Table 7.2 - *TREC-4* Accumulator Timings (in Seconds).

Table 7.2 illustrates the improvement in the time taken to process each query batch with an increasing maximum accumulator number. When the timing values are brought down to a per query basis we see a 61.77% reduction in the time to process a query using 5,000 accumulators as opposed to 117,500 accumulators. This improvement does come with a reduction in effectiveness in terms of relevant documents retrieved (-15.07%) and in terms of average precision (-5.26%). However these figures apply to the most restrictive accumulator value. If this restriction is relaxed to allow 22,500 accumulators to be active per query, then once again we have a totally different situation. We still achieve a large drop (-48.55%) in the average time taken to process a query but the corresponding drop in effectiveness in term of number of relevant documents returned (-0.32%) is minimal and average precision is almost the same (+0.01%).

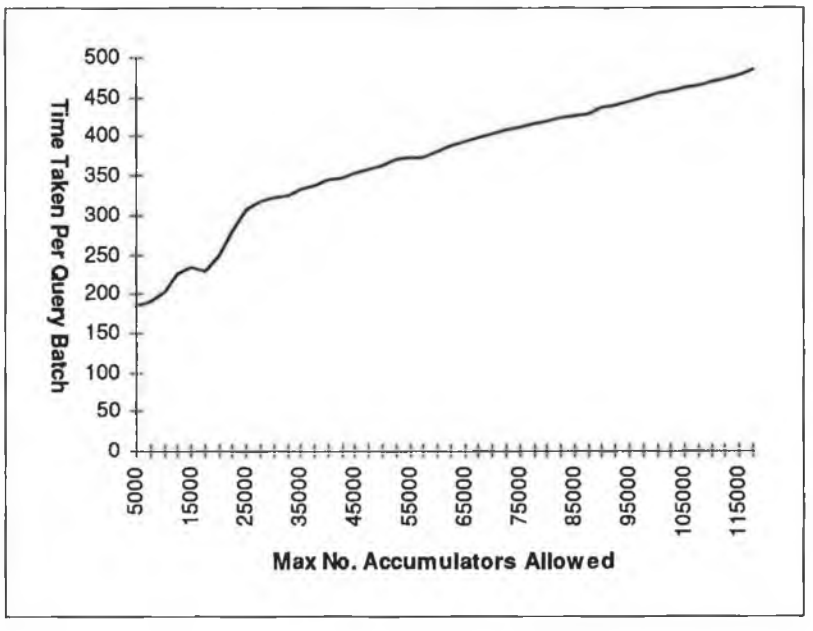


Figure 7.19 - *TREC-4* Time Taken Vs Accumulators Used.

As with the *TREC-3* sub-collection we normalised and combined the effectiveness graphs with the efficiency graph (Figure 7.20). The graphs in Figure 7.20 illustrate the amount of efficiency improvements that can be achieved without any degradation of effectiveness. The benefit to the IR engine can be measured as the area between the solid line (effectiveness) and the dashed line (efficiency) in both graphs in Figure 7.20. System effectiveness is maintained up until the accumulator restriction is set below 20,000 accumulators per query.

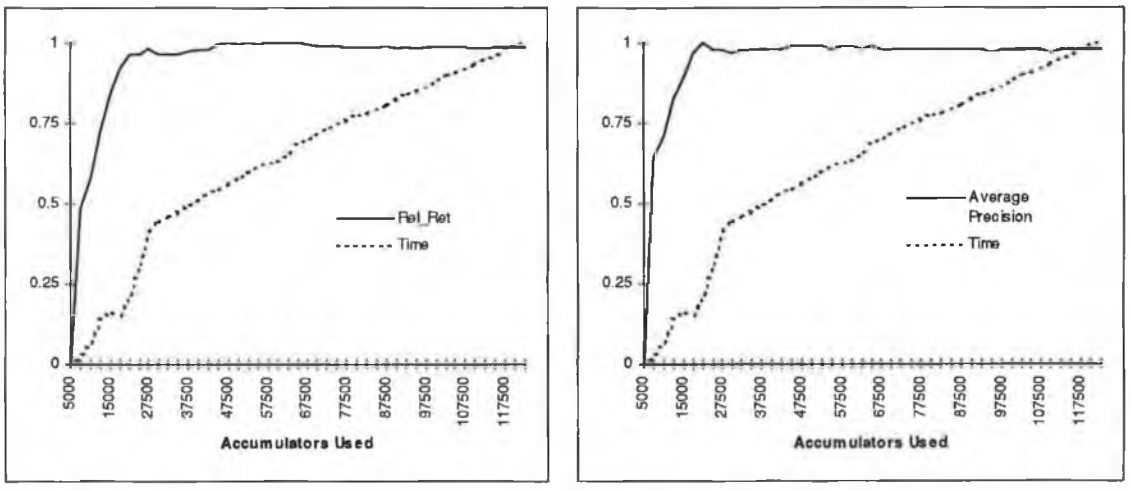


Figure 7.20 - *TREC-4* Accumulator Efficiency Vs Effectiveness.

When Figure 7.4 and Figure 7.20 are compared certain similarities can be drawn in that it is most definitely possible to significantly reduce the amount of accumulators activated per query without compromising effectiveness. There is a subtle difference between Figure 7.4 and Figure 7.20 however. In Figure 7.4 it is possible to reduce the number of accumulators activated to ~35,000 without compromising query effectiveness. In Figure 7.20 however the reduction in query effectiveness does not come into play until the number of accumulators allowed activate per query is reduced to ~22,500. This is largely due to the average query length of the *TREC-3* and *TREC-4* query sets. The *TREC-4* query set is on average significantly smaller than the *TREC-3* query set. This would suggest that it is possible to roughly predict where the maximum limit on accumulator activation should be placed by taking into account the query length (number of indexable search terms).

7.5.2 Query Term Thresholding in *TREC-4*.

This section details the *QTT* experiments carried out on the *TREC-4* collection. It must be remembered that the *TREC-4* query set were on average much smaller in terms of the number of query terms per query and as such this should and indeed does have an impact on the performance of the *QTT* approach. It can be seen from Figure 7.21 and Figure 7.22 that the effectiveness curves flatten out very quickly as the *QTT* percentage increases. When this happens it means that the *QTT* is no longer having an effect on the processing of the query because all of the query is falling within the *QTT* setting, i.e. all of the query is being processed as with the *TREC-3* experiments.

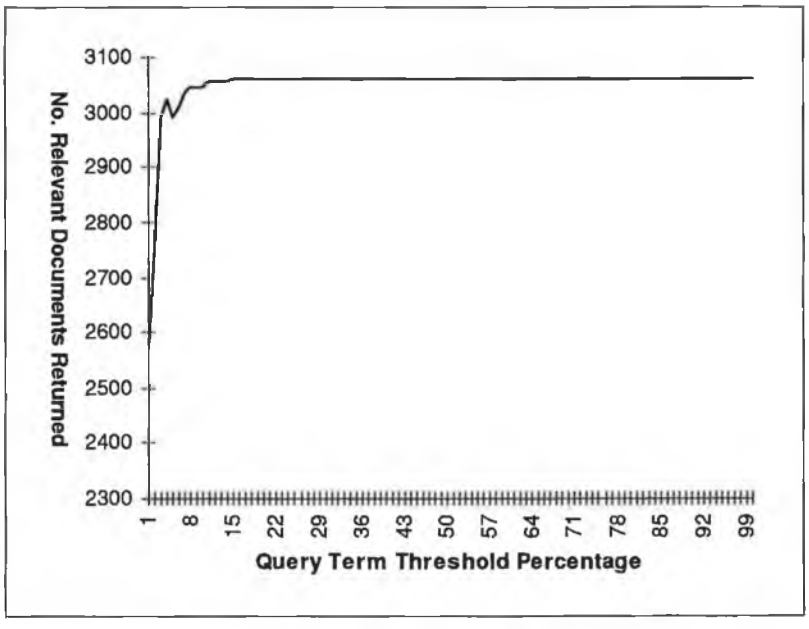


Figure 7.21 - *TREC-4 QTT* Percentage Vs Relevant Documents.

This is due to the short nature of the *TREC-4* query set. It also lessens somewhat the potential for achieving efficiency improvements while maintaining effectiveness. However that potential is still there and should be exploited if at all possible. Figure 7.23 shows the effect on the efficiency of the retrieval operation for varying *QTT* settings. It shows that for *QTT* settings of up to 25% savings in efficiency can be obtained. After this point all of the terms in 'short' queries are being processed anyway and the possibility for efficiency improvements using this approach disappear.

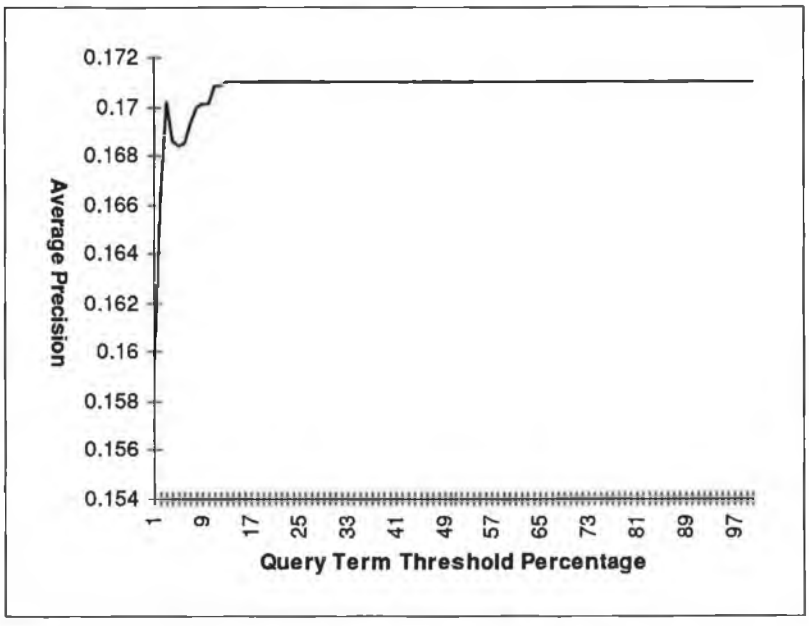
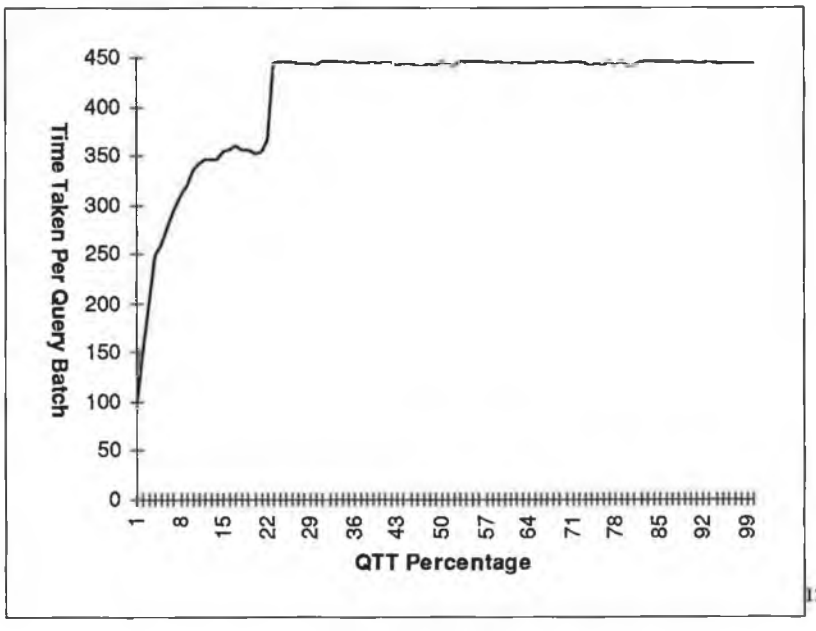


Figure 7.22 - *TREC-4 QTT* Percentage Vs Average Precision.



[12]

Figure 7.23 - *TREC-4* Time Taken Vs *QTT* Percentage.

Figure 7.24 illustrates the potential saving attainable. In both graphs the efficiency curve falls significantly before there is a corresponding drop in the

---

[12] Undulations in the graph when it flattens out are due to slight variations of system execution time detected by the UNIX 'time' command.

effectiveness curve indicating the potential for efficiency improvements without compromising effectiveness.

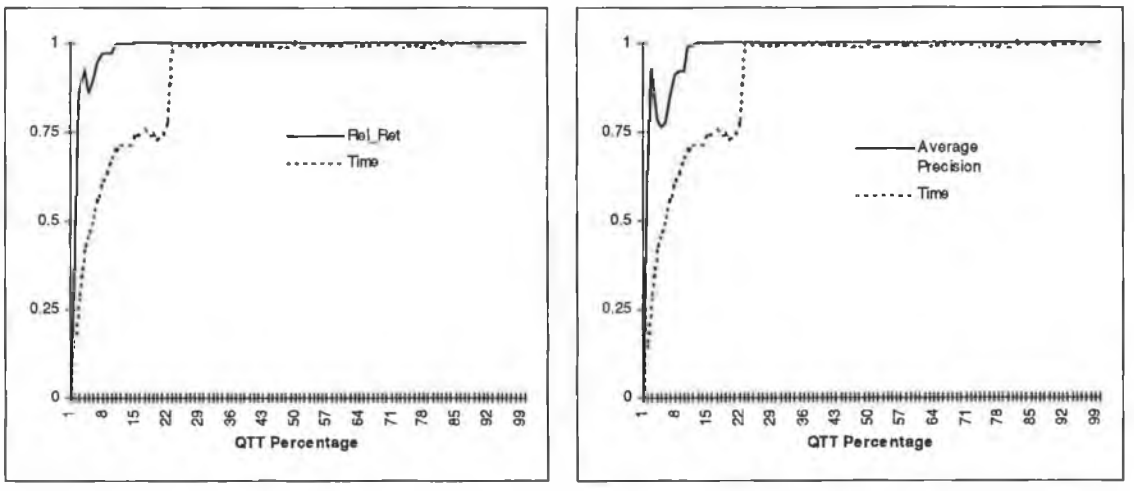


Figure 7.24 - *TREC-4 QTT* Efficiency Vs Effectiveness.

7.5.3 Thresholding Combinations in *TREC-4*.

As with the *TREC-3* experiments each of the thresholding approaches has been evaluated in isolation. It is now necessary to assess the impact of the combination of thresholding approaches on the *TREC-4* collection. The same set of thresholding combination experiments carried out on the *TREC-3* collection was also carried out on the *TREC-4* collection.

	QSR
Retrieved:	50000
Relevant:	6501
Rel_ret:	**3062**
P. at 0.0	0.5305
P. at 0.1	0.3586
P. at 0.2	0.3031
P. at 0.3	0.2564
P. at 0.4	0.2057
P. at 0.5	0.1512
P. at 0.6	0.1212
P. at 0.7	0.0742
P. at 0.8	0.0469
P. at 0.9	0.0222
P. at 1.0	0.0016
Av. P	**0.1709**

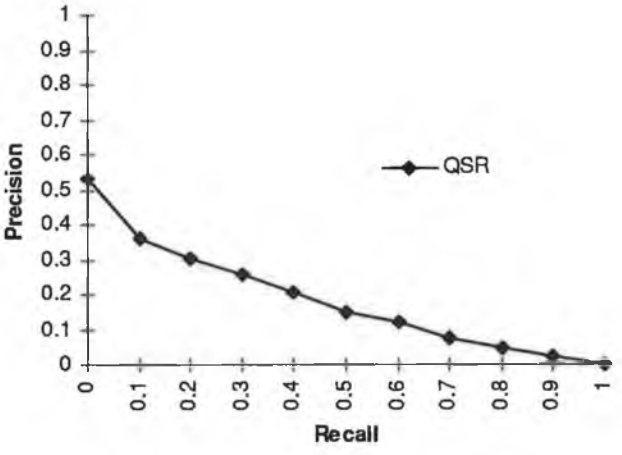


Figure 7.25 - Optimal Effectiveness Performance for *TREC-4* collection.

The optimal settings for the *TREC-4* collection are detailed in Figure 7.26. The optimal system settings for the *TREC-4* collection are almost identical to the optimal

settings for the *TREC-3* collection. The only difference is the weighting scheme used which does not take into account the frequency of occurrence of the query terms within the query.

Parameter	Optimal Value
Query Term Thresholding:	6%
Posting List Thresholding:	*(Start)* 10%, *(End)* 90%
Maximum Document Accumulators:	50,000
Weighting Scheme:	log(tf) * (IDF^2)
Document Normalisation Scheme:	No Normalisation
Document Fragmentation:	On *(Page Size 200 keywords)*

Figure 7.26 - Optimal System Parameter Settings (*TREC-4*).

7.6 TREC-5 Experimental Settings.

In order to test the validity of the experimental results obtained using the *TREC-3* and *TREC-4* collections with respect to the thresholding settings used for Accumulator Restriction, *QTT* and *PLT* we used the settings which achieved optimal performance for the *TREC-3* and *TREC-4* collections on the TREC-5 collection. The results of which will be presented in Chapter 8.

7.7 Summary.

The experiments presented in this Chapter represent only a small portion of the total number of experiments carried out using our IR engine. The experiments presented are intended to illustrate the benefits of using the combination of a modified inverted file structure coupled with judicious and appropriate thresholding combinations. This system has been used in all of the official submissions from Dublin City University to *TREC-5*. In the next Chapter we will present our conclusions based on the outcome of the experiments detailed in this Chapter along with their potential implications for future research in this area.

8. Conclusions.

8.1 Introduction.

In this Chapter we will discuss the experimental results obtained in Chapter 7 in greater detail, paying particular attention to the results obtained for the *TREC-5* sub-collection. As already stated in Chapter 7 the experiments carried out on the *TREC-3* and *TREC-4* sub-collections were training runs carried out in order to fine tune the various parameters of the thresholding techniques. Once fine tuned, these parameter settings were used blind in the *TREC-5* collection's experimental set. This meant we submitted *TREC-5* runs not knowing relevance assessments in advance and our top-ranked documents were judged. In all four official experimental runs were carried out on the *TREC-5* collection, two using automatically generated queries and two using manually generated queries. In one of the automatic runs and one of the manual runs the QSR techniques were switched on (using the parameter settings obtained from the *TREC-3* and *TREC-4* training runs) and the other automatic and manual runs used no QSR techniques whatsoever.

In this Chapter we will analyse on a per query basis the results obtained for the *TREC-5* collection and determine the differences (if any) in terms of effectiveness and efficiency between the two automatic runs (one with QSR and one without) and the two manual runs (one with QSR and one without). For the purposes of this thesis automatic runs can be interpreted as short queries (with an average of 7.96 terms per query) and manual runs as longer queries (with an average of 29.9 terms per query)

8.2 QSR Setting used for TREC-5 Experiments.

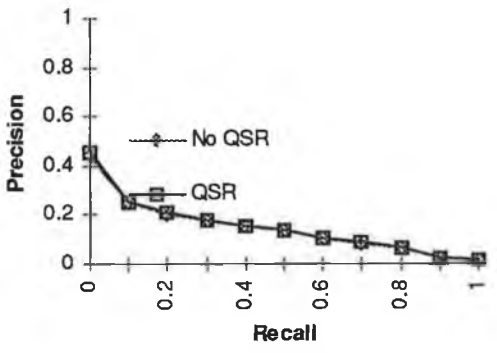
As the experiments detailed in Chapter 7 illustrate we used the *TREC-3* and *TREC-4* collections to evaluate the performance of our QSR techniques on realistic test collections. With each collection having different characteristics with respect to the type of document being indexed and the type of queries being applied to the system. After extensive testing and evaluation of our system's performance on both collections optimal settings were recorded for each collection. These settings (detailed in Sections

7.4.4 and 7.5.3) were then applied *'blind'* to the new *TREC-5* collection in order to test the validity of the results obtained to date using the *TREC-3* and *TREC-4* collections.

8.3 Overall Performance of Automatic and Manual Runs.

Figure 8.1 details the performance of our QSR techniques versus an approach using no QSR techniques for our automatic submission to the *TREC-5* conference. It can be seen from Figure 8.1 that there is no performance degradation with respect to system effectiveness when our QSR techniques are employed. In fact there are slight improvements in the Average precision (from .1334 to .1340) and the number of relevant documents returned (from 1940 to 1943). The real and by far the most notable impact of employing our QSR techniques is in the area of system efficiency. When our QSR techniques are in operation we obtain a 48.57% decrease in the time taken to process our automatic *TREC-5* submission (50 queries) from 451.1 seconds to 232.2 seconds.

	No QSR	QSR
Retrieved:	50000	50000
Relevant:	5524	5524
Rel_ret:	**1940**	**1943**
P. at 0.0	0.4404	0.4525
P. at 0.1	0.2531	0.254
P. at 0.2	0.2067	0.2076
P. at 0.3	0.1802	0.1805
P. at 0.4	0.1571	0.1575
P. at 0.5	0.138	0.1374
P. at 0.6	0.1085	0.1079
P. at 0.7	0.0843	0.0878
P. at 0.8	0.0639	0.0644
P. at 0.9	0.0275	0.0277
P. at 1.0	0.0192	0.0181
Av. P	**0.1334**	**0.134**
P @ 10 D.	0.246	0.254
P @ 30 D.	0.188	0.187
P @ 100 D.	0.124	0.124



	No QSR	QSR	% Red.
Seconds:	451.1	232.2	48.57%
Doc. Acc:	13,823,647	240,595	82.60%
Postings:	17,479,134	8,714,946	50.14%

Figure 8.1 - Performance comparison of QSR Vs No QSR for Automatic Run.

This reduction in the time taken to process the queries is obtained from reducing the amount of the Query Space processed from around 17.5 million posting entries to 8.7 million posting entries (a reduction of 50.14%%) and also reducing the total number of active document accumulators which require sorting in order to provide the user with a ranked list of documents from nearly 14 million accumulators to just under

2.5 million accumulators (a reduction of 82.60%). The improvements in system efficiency result in a real and very noticeable reduction in system response time to user's queries even when the queries themselves are short.

Appendix A details, graphically and in tabular format, the impact of our QSR techniques on our automatic *TREC-5* submission on a per query basis. Table 8.1 summarises on a per query basis the system response time firstly using no QSR and secondly using QSR. It can be seen from Table 8.1 that the improvements obtained by using QSR vary quite considerably (in this instance from no improvement whatsoever to just over 70%). This is due to a large degree on the size of the query being processed by the system, the smaller the query the less room there is for reducing its QS. The number of possible active document accumulators depends more on the specificity of the terms in the query and less on the actual size of the query i.e. a larger query composed of highly selective terms will tend to activate less document accumulators than a small query containing very commonly occurring terms.

Query	No QSR	QSR	% Red.
251	12.4	5.8	53.23%
252	10.1	4.5	55.45%
253	12.2	4	67.21%
254	8.3	5.4	34.94%
255	10.9	5	54.13%
256	16.9	5.5	67.46%
257	9.8	3.7	62.24%
258	9.4	5.3	43.62%
259	9.2	4.8	47.83%
260	8.9	4.5	49.44%
261	15.3	8.6	43.79%
262	5.7	3.9	31.58%
263	5.5	3.7	32.73%
264	7.9	5	36.71%
265	3.1	3.1	0.00%
266	4.6	3.6	21.74%
267	14.6	6.3	56.85%
268	11.6	4	65.52%
269	9	5.1	43.33%
270	8.7	4.7	45.98%
271	10	5.1	49.00%
272	2.5	2.4	4.00%
273	9.9	4.5	54.55%
274	8.9	5.1	42.70%
275	10.3	5	51.46%
276	7.6	5.3	30.26%
277	11.5	4.4	61.74%
278	8.3	4.2	49.40%
279	2.9	2.8	3.45%
280	9	3.8	57.78%
281	7.8	4.2	46.15%
282	2.9	2.8	3.45%
283	10.3	4.8	53.40%
284	8.3	5.7	31.33%
285	11.2	4	64.29%
286	7.8	3.8	51.28%
287	5.2	3.6	30.77%
288	5.2	3.8	26.92%
289	12.8	5.4	57.81%
290	11.6	4.8	58.62%
291	13.7	4.1	70.07%
292	12.9	5.2	59.69%
293	3.9	3.5	10.26%
294	10.1	7.5	25.74%
295	2.6	2.5	3.85%
296	11.8	7.2	38.98%
297	2.1	2.1	0.00%
298	9.7	4.7	51.55%
299	12	5.7	52.50%
300	14.2	7.5	47.18%
Totals:	451.1	232	48.57%

Table 8.1 - Time per Query (in Seconds) QSR Vs No QSR (Automatic Run).

This results in a situation in which the potential for achieving reductions in the system response time depends more on the size of the QS and less on the possible size of the active document accumulator set. Table 8.2 summarises on a per query basis the

number of document accumulators activated on a per query basis firstly using no QSR and secondly using QSR. It can be seen from Table 8.2 that an upper limit of 50,000 accumulators per query was used when QSR was in operation and that not all queries reached this upper limit. The queries that did not reach the upper accumulator limit tend to have a small number of query terms. In these situations their is no potential document accumulator savings to be gained as there is no reduction in the active accumulator set. The improvements in the number of document accumulators activated range from just over 2% to over 90%.

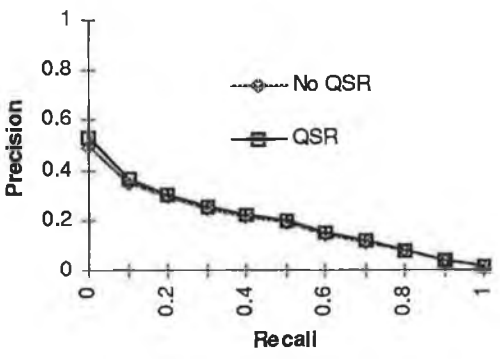
Query	No QSR	QSR	% Red.	Query	No QSR	QSR	% Red.
251	352,015	50,000	85.80%	276	204,823	50,000	75.59%
252	255,637	50,000	80.44%	277	344,358	50,000	85.48%
253	363,934	50,000	86.26%	278	273,950	50,000	81.75%
254	199,519	50,000	74.94%	279	40,143	38,026	5.27%
255	432,511	50,000	88.44%	280	301,021	50,000	83.39%
256	500,000	50,000	90.00%	281	238,818	50,000	79.06%
257	390,482	50,000	87.20%	282	38,157	36,795	3.57%
258	303,625	50,000	83.53%	283	293,952	50,000	82.99%
259	333,427	50,000	85.00%	284	250,223	50,000	80.02%
260	231,237	50,000	78.38%	285	453,323	50,000	88.97%
261	419,072	50,000	88.07%	286	222,254	50,000	77.50%
262	97,498	50,000	48.72%	287	105,784	50,000	52.73%
263	117,661	50,000	57.51%	288	112,573	50,000	55.58%
264	202,846	50,000	75.35%	289	481,713	50,000	89.62%
265	51,269	49,358	3.73%	290	414,176	50,000	87.93%
266	92,701	50,000	46.06%	291	434,140	50,000	88.48%
267	434,766	50,000	88.50%	292	493,289	50,000	89.86%
268	406,709	50,000	87.71%	293	60,516	50,000	17.38%
269	291,815	50,000	82.87%	294	278,401	50,000	82.04%
270	267,851	50,000	81.33%	295	26,432	25,347	4.10%
271	377,593	50,000	86.76%	296	294,735	50,000	83.04%
272	31,062	29,610	4.67%	297	27,004	26,459	2.02%
273	303,018	50,000	83.50%	298	371,754	50,000	86.55%
274	303,364	50,000	83.52%	299	432,024	50,000	88.43%
275	370,730	50,000	86.51%	300	499,742	50,000	89.99%
				Totals:	13,823,647	2,405,955	82.60%

Table 8.2 - Active Accumulators (QSR Vs No QSR) Automatic.

Figure 8.2 details the performance of our QSR techniques versus no QSR for our manual *TREC-5* submission. The overall performance in terms of effectiveness is better for the manual runs than the automatic runs due primarily to the more descriptive nature of the manually formulated queries. However when we compare the manual run using and not using QSR with each other we again see no performance degradation in system effectiveness, in fact as with the automatic TREC-5 submissions there is a slight improvement in the average precision (.1804 to .1862) and only a small reduction in the total number of relevant documents returned (2472 to 2384) when our QSR techniques are employed. As with the automatic TREC-5 submissions there are significant improvements in system efficiency with the total time taken to process the

set of 50 queries dropping from 1122.1 seconds to 411.9 (a reduction of 63.29%). This improvement in system response time is achieved through reducing the amount of the Query Space processed from around 63.6 million posting entries to just over 21 million posting entries (a reduction of 66.99%) and reducing the total number of activated document accumulators which require sorting in order to provide the user with a ranked list of documents from around 31.5 million to 2.5 million (a reduction of 92.07%).

	No QSR	QSR
Retrieved:	50000	50000
Relevant:	5524	5524
Rel_ret:	**2472**	**2384**
P. at 0.0	0.4952	0.5336
P. at 0.1	0.3507	0.3615
P. at 0.2	0.2897	0.2979
P. at 0.3	0.2492	0.2571
P. at 0.4	0.2171	0.2254
P. at 0.5	0.1883	0.1955
P. at 0.6	0.1437	0.1478
P. at 0.7	0.1136	0.1173
P. at 0.8	0.0756	0.0814
P. at 0.9	0.0367	0.0403
P. at 1.0	0.0177	0.0181
Av. P	**0.1804**	**0.1862**
P @ 10 D.	0.316	0.332
P @ 30 D.	0.2427	0.2547
P @ 100 D.	0.166	0.1678



	No QSR	QSR	% Red.
Seconds:	1122.1	411.9	63.29%
Doc. Acc:	31,552,542	2,500,000	92.07%
Postings:	63,643,838	21,005,921	66.99%

Figure 8.2 - Performance comparison of QSR Vs No QSR for Manual Run.

Again it can be seen that significant improvements in system efficiency are attained without impacting on system effectiveness for longer and more descriptive user queries. Appendix B details, graphically and in tabular format, the impact of our QSR techniques on our manual *TREC-5* submission on a per query basis. Figure 8.3 summarises on a per query basis the system response time firstly using no QSR and secondly using QSR for our manual *TREC-5* submission. Unlike the automatic *TREC-5* submission there is much less variability in the improvements obtained in system response time with the minimum improvement being around 38% and the maximum improvement being just over 74%. This is primarily due to the larger average QS size which results in a the possibility of greater potential savings.

Query	No QSR	QSR	% Red.
251	14.7	4.9	66.67%
252	14.6	7.7	47.26%
253	24.8	10.2	58.87%
254	13.1	8	38.93%
255	20.6	8	61.17%
256	23.6	8.3	64.83%
257	15.7	5.3	66.24%
258	40.3	13.1	67.49%
259	14	4.5	67.86%
260	18.6	8.6	53.76%
261	43.9	19.4	55.81%
262	22.7	11	51.54%
263	34.1	8.8	74.19%
264	18.6	7.4	60.22%
265	16.8	7.4	55.95%
266	9.8	4.5	54.08%
267	29.8	10.8	63.76%
268	21.7	7.4	65.90%
269	16.3	7.4	54.60%
270	41.7	10.9	73.86%
271	19	6.7	64.74%
272	22.3	8.4	62.33%
273	16	6.1	61.88%
274	15.4	4.8	68.83%
275	30.7	10.7	65.15%

Query	No QSR	QSR	% Red.
276	20	8.7	56.50%
277	14.7	6.4	56.46%
278	17.2	5.7	66.86%
279	15.5	7.3	52.90%
280	12.1	4.5	62.81%
281	20	7.3	63.50%
282	11.5	5.5	52.17%
283	32.5	9.5	70.77%
284	21.5	8.5	60.47%
285	31.2	9.1	70.83%
286	18.9	6	68.25%
287	27.6	8.6	68.84%
288	31.9	12.6	60.50%
289	44.4	13.1	70.50%
290	19.7	7.1	63.96%
291	34.8	10.1	70.98%
292	24.6	8.3	66.26%
293	21.8	6.9	68.35%
294	15.6	9	42.31%
295	13.2	6	54.55%
296	16.3	6	63.19%
297	35	11.8	66.29%
298	17.8	6.3	64.61%
299	26.2	8.8	66.41%
300	19.3	8.5	55.96%
Totals:	1122.1	411.9	63.29%

Figure 8.3 - Time per Query (in Seconds) QSR Vs No QSR (Manual Run).

As expected the number of active document accumulators is much larger for the manual *TREC-5* submission, this simply due to the larger average query size.

Query	No QSR	QSR	% Red.
251	480,613	50,000	89.60%
252	495,791	50,000	89.92%
253	728,024	50,000	93.13%
254	386,854	50,000	87.08%
255	662,655	50,000	92.45%
256	752,355	50,000	93.35%
257	582,033	50,000	91.41%
258	850,912	50,000	94.12%
259	461,730	50,000	89.17%
260	573,533	50,000	91.28%
261	849,810	50,000	94.12%
262	684,119	50,000	92.69%
263	796,261	50,000	93.72%
264	605,513	50,000	91.74%
265	568,257	50,000	91.20%
266	297,520	50,000	83.19%
267	739,358	50,000	93.24%
268	684,250	50,000	92.69%
269	531,714	50,000	90.60%
270	863,920	50,000	94.21%
271	619,386	50,000	91.93%
272	681,675	50,000	92.67%
273	491,507	50,000	89.83%
274	489,878	50,000	89.79%
275	770,324	50,000	93.51%

Query	No QSR	QSR	% Red.
276	590,936	50,000	91.54%
277	492,323	50,000	89.84%
278	576,267	50,000	91.32%
279	502,543	50,000	90.05%
280	415,334	50,000	87.96%
281	613,260	50,000	91.85%
282	368,555	50,000	86.43%
283	788,228	50,000	93.66%
284	636,520	50,000	92.14%
285	799,602	50,000	93.75%
286	607,649	50,000	91.77%
287	704,911	50,000	92.91%
288	808,639	50,000	93.82%
289	855,010	50,000	94.15%
290	645,020	50,000	92.25%
291	809,266	50,000	93.82%
292	738,092	50,000	93.23%
293	709,656	50,000	92.95%
294	460,013	50,000	89.13%
295	409,572	50,000	87.79%
296	536,148	50,000	90.67%
297	814,654	50,000	93.86%
298	599,743	50,000	91.66%
299	785,072	50,000	93.63%
300	637,537	50,000	92.16%
Totals:	31,552,542	250,000	92.08%

Figure 8.4 - Active Accumulators (QSR Vs No QSR) Manual.

Figure 8.4 details the reduction in document accumulators activated on a per query basis. It can be seen from Figure 8.4 that when QSR is switched on all queries hit the upper bound of allowable active accumulators which results in improvements in system efficiency due to a much smaller number of accumulators being passed to the procedure which presents the user with a ranked list of documents. The reduction in the number of active document accumulators for the manual *TREC-5* submission is much more consistent, varying only from a minimum of around 83% to a maximum of just over 94%. This is simply attributed to the larger manual queries activating a lot more document accumulators than the automatic *TREC-5* submission's query set. The difference in the size of the automatic and manual query sets is not only reflected in the system response time but also in the size of the active document accumulator sets for the automatic and manual query sets and the average size of the Query Spaces constructed during retrieval.

	Automatic No QSR	QSR	% Red	Manual No QSR	QSR	% Red
Average Response Time(Seconds)	9.022	4.644	48.57%	24.22	8.238	65.98%
Average. No. Doc. Accumulators	276,472	48,119	82.60%	631,050	50,000	92.07%
Average No. Postings Processed	349,582	174,298	50.14%	1,272,876	420,118	66.99%

Table 8.3 - Average Efficiency Measures for Automatic and Manual Submissions.

Table 8.3 presents the average values for the three efficiency criteria used to evaluate the performance of the system, which are the average response time to a users query, the average number of active document accumulators and the average number of posting entries processed during retrieval. It is improvements in the second two criterion which result in improvements in the system response time. The difference in the amount of savings obtained between the automatic submission and the manual submission can be largely attributed to the difference in the query sets used for the automatic and manual submissions. This difference is summarised in Table 8.4.

	Automatic Total	Min	Max	Average	Manual Total	Min	Max	Average
Terms	398	2	22	7.96	1495	9	87	29.9

Table 8.4 - Comparison of Automatic and Manual Query Sets.

The information presented in Table 8.4 summarises the internal system representation of the entire set of 50 automatically formulated queries and 50 manually formulated queries and as such includes phrases but not stopwords. A more detailed description of the contents of the internal system representation of each query can be obtained in Appendix A (automatically generated queries) and in Appendix B (manually generated queries).

Table 8.5 details on a per query basis the effect of using QSR on the number of relevant documents returned to the user in response to a query. Table 8.6 details on a per query basis the effect of using QSR on the average precision of a query, with 'S' meaning that the QSR and No QSR approaches have the same values, 'I' meaning that using QSR has resulted in an improvement and 'D' meaning that using QSR has resulted in a dis-improvement in the number of relevant documents returned in response to a query.

Query	No QSR	QSR	S	I	D
251	52	52	✓		
252	18	18	✓		
253	8	8	✓		
254	39	39	✓		
255	21	20			✓
256	13	13	✓		
257	83	83	✓		
258	45	46		✓	
259	31	31	✓		
260	8	11		✓	
261	55	55	✓		
262	4	4	✓		
263	15	15	✓		
264	51	51	✓		
265	136	136	✓		
266	25	25	✓		
267	1	1	✓		
268	6	11		✓	
269	20	12			✓
270	42	42	✓		
271	60	60	✓		
272	29	29	✓		
273	289	303		✓	
274	39	39	✓		
275	9	8			✓
276	6	6	✓		
277	48	48	✓		
278	1	1	✓		
279	2	2	✓		
280	32	31			✓
281	1	1	✓		
282	57	57	✓		
283	42	41			✓
284	24	20			✓
285	178	183		✓	
286	90	88			✓
287	23	23	✓		
288	77	77	✓		
289	26	26	✓		
290	21	20			✓
291	30	28			✓
292	3	3	✓		
293	13	13	✓		
294	21	21	✓		
295	12	12	✓		
296	0	0	✓		
297	26	26	✓		
298	60	62		✓	
299	33	27			✓
300	15	15	✓		
		Totals:	34	6	10

Table 8.5 - Changes in Relevant Documents Returned per Automatic Query.

It can be seen from Table 8.5 that the number of relevant documents returned remains unaffected in the majority of the 50 queries, improvements in the number of relevant documents returned are obtained for 6 queries, and reductions in the number of relevant documents returned for 10 queries. The same general performance holds

true for the average precision values (Table 8.6). Again we have 34 queries in which there is no change in the average precision value. There are improvements in average precision for 4 queries and dis-improvements for 12 queries.

Query	No QSR	QSR	S	I	D
251	0.0051	0.0042			✓
252	0.0286	0.0328	✓		
253	0.7764	0.7764	✓		
254	0.0527	0.0531	✓		
255	0.0075	0.0067			✓
256	0.0518	0.0544	✓		
257	0.1923	0.1993		✓	
258	0.031	0.0337	✓		
259	0.294	0.2954	✓		
260	0.0026	0.0044	✓		
261	0.1893	0.1905	✓		
262	0.5429	0.5429	✓		
263	0.1968	0.1974	✓		
264	0.0167	0.0171	✓		
265	0.6598	0.6595			✓
266	0.0161	0.0163	✓		
267	0.0011	0.0013	✓		
268	0.001	0.0035	✓		
269	0.0007	0.0003			✓
270	0.0509	0.055	✓		
271	0.1423	0.1446	✓		
272	0.1973	0.2026		✓	
273	0.1884	0.2178		✓	
274	0.0289	0.0279			✓
275	0.0136	0.0136	✓		

Query	No QSR	QSR	S	I	D
276	0.6937	0.6937	✓		
277	0.2625	0.2542			✓
278	0.0026	0.0025			✓
279	0.2083	0.2083	✓		
280	0.5604	0.5653	✓		
281	0.0042	0.0046	✓		
282	0.0998	0.1	✓		
283	0.0633	0.0677	✓		
284	0.0344	0.0336			✓
285	0.1756	0.189		✓	
286	0.2485	0.1934			✓
287	0.0475	0.0483	✓		
288	0.3134	0.3122			✓
289	0.0333	0.0373	✓		
290	0.0051	0.0046			✓
291	0.0032	0.004	✓		
292	0.0003	0.0003	✓		
293	0.0042	0.0042	✓		
294	0.0176	0.0177	✓		
295	0.0887	0.0893	✓		
296	0	0	✓		
297	0.0093	0.0093	✓		
298	0.0544	0.056	✓		
299	0.0309	0.0246			✓
300	0.0227	0.0294	✓		
		Totals:	34	4	12

Table 8.6 - Changes in Average Precision per Automatic Query.

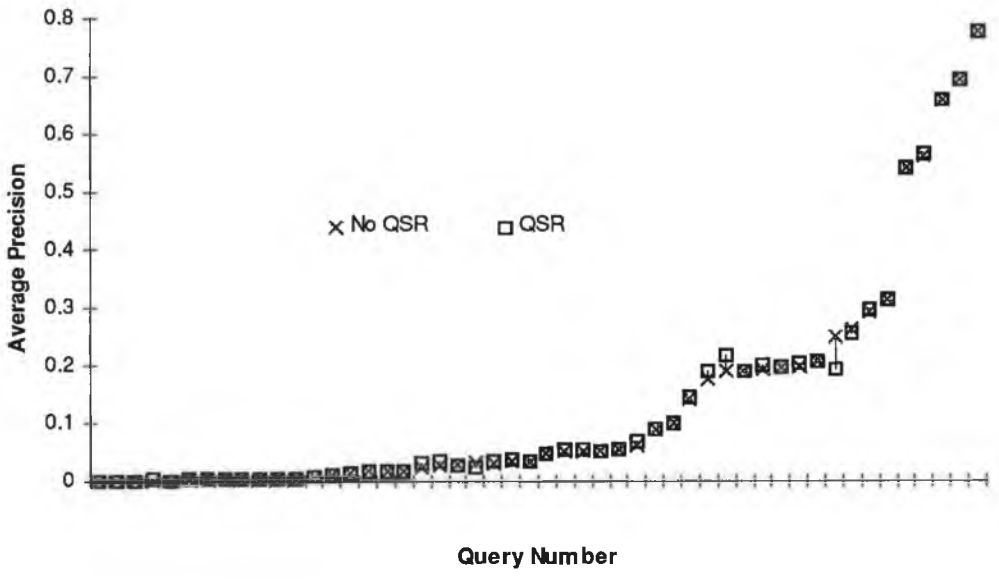


Figure 8.5 - Changes in Avg Precision sorted by increasing Avg Precision (Auto)

Query	No QSR	QSR	S	I	D
251	69	51			✓
252	28	28	✓		
253	9	9	✓		
254	47	48		✓	
255	30	32		✓	
256	10	12		✓	
257	105	104			✓
258	69	56			✓
259	33	34		✓	
260	14	15		✓	
261	71	71	✓		
262	4	4	✓		
263	15	15	✓		
264	39	33			✓
265	140	140	✓		
266	87	87	✓		
267	1	1	✓		
268	13	7			✓
269	34	27			✓
270	87	87	✓		
271	74	76		✓	
272	30	31		✓	
273	261	282		✓	
274	64	62			✓
275	14	14	✓		

Query	No QSR	QSR	S	I	D
276	6	6	✓		
277	44	52		✓	
278	3	3	✓		
279	2	2	✓		
280	32	31			✓
281	1	1	✓		
282	64	65		✓	
283	53	48			✓
284	35	32			✓
285	236	244		✓	
286	94	73			✓
287	24	25		✓	
288	78	78	✓		
289	55	58		✓	
290	10	13		✓	
291	116	56			✓
292	30	31		✓	
293	9	9	✓		
294	42	42	✓		
295	11	11	✓		
296	0	0	✓		
297	63	64		✓	
298	70	67			✓
299	27	27	✓		
300	19	20		✓	
		Totals:	20	17	13

Table 8.7 - Changes in Relevant Documents Returned per Manual Query.

Table 8.7 and Table 8.8 detail on a per query basis the changes in the number of relevant documents returned and average precision when QSR is switched on for the manually constructed *TREC-5* query set.

Query	No QSR	QSR	S	I	D
251	0.0102	0.006			✓
252	0.0651	0.0661	✓		
253	0.7767	0.7939		✓	
254	0.1365	0.1453		✓	
255	0.0152	0.0165	✓		
256	0.0083	0.0103	✓		
257	0.1309	0.1296			✓
258	0.0834	0.0715			✓
259	0.5392	0.5419	✓		
260	0.0525	0.0662		✓	
261	0.3257	0.3301	✓		
262	0.5012	0.525		✓	
263	0.1607	0.1968		✓	
264	0.0068	0.0044			✓
265	0.7151	0.7095			✓
266	0.1411	0.1413	✓		
267	0.0007	0.0007	✓		
268	0.0082	0.0046			✓
269	0.0025	0.0014			✓
270	0.4559	0.4856		✓	
271	0.3471	0.3572		✓	
272	0.2238	0.2421		✓	
273	0.1458	0.1742		✓	
274	0.0979	0.0996	✓		
275	0.021	0.0244	✓		

Query	No QSR	QSR	S	I	D
276	0.6321	0.6321	✓		
277	0.2456	0.2582		✓	
278	0.0274	0.0208			✓
279	0.1667	0.1667	✓		
280	0.614	0.6334		✓	
281	0.0154	0.0149			✓
282	0.1148	0.1161	✓		
283	0.0481	0.0471			✓
284	0.0743	0.1005		✓	
285	0.5336	0.5398		✓	
286	0.243	0.1788			✓
287	0.0344	0.0674		✓	
288	0.3984	0.4124		✓	
289	0.0977	0.1256		✓	
290	0.0013	0.0017	✓		
291	0.038	0.0101			✓
292	0.0147	0.0173	✓		
293	0.002	0.002	✓		
294	0.0529	0.0533	✓		
295	0.0866	0.0915	✓		
296	0	0	✓		
297	0.4166	0.4894		✓	
298	0.1554	0.1407			✓
299	0.0181	0.0213	✓		
300	0.0179	0.0224	✓		
		Totals:	20	17	13

Table 8.8 - Changes in Average Precision per Manual Query.

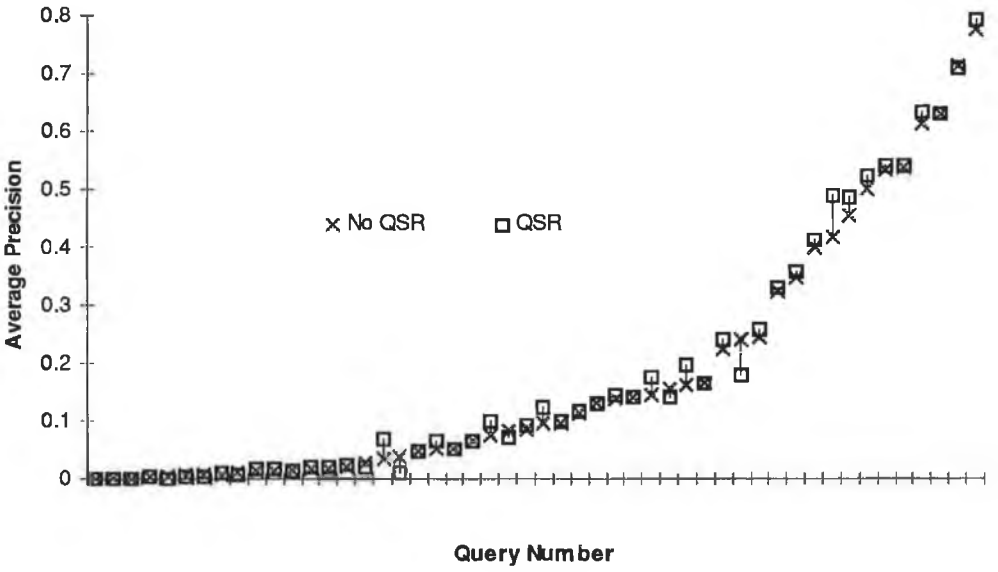


Figure 8.6 - Changes in Avg Precision sorted by increasing Avg Precision (Man).

It can be seen from Table 8.7 and Table 8.8 that 20 of the queries remain unaffected with improvements obtained for 17 query and dis-improvements for 13 when QSR is switched on. In the majority of cases the improvements or dis-improvements in system effectiveness with respect to the number of relevant documents returned and average precision are very small and in all cases the corresponding improvement in system efficiency is significant.

8.4 Experimental Conclusions.

This Chapter presented the results obtained using our QSR techniques on the *TREC-5* collection. The QSR settings used for the *TREC-5* collection were determined by previous experimentation on the *TREC-3* and *TREC-4* collections. Two sets of experiments were carried out on the *TREC-5* collection, automatically generated queries with an average length of 7.96 query terms and manually constructed query terms with an average length of 29.9 query terms. We applied the optimally performing QSR settings of the *TREC-3* collection (in particular its queries) to the manually constructed *TREC-5* queries because the characteristics of the query sets were similar. The characteristics of the *TREC-4* collection's query set are similar to the automatically generated *TREC-5* query set so we used the optimal QSR settings from the *TREC-4* experiments for the *TREC-5* automatic submission.

We achieved consistent results from both the automatic and manual *TREC-5* submissions in that we incurred virtually no performance degradation with respect to system effectiveness while achieving significant performance improvements with respect to system efficiency. The amount of improvement in efficiency obtained for the manual *TREC-5* submission was greater than the level obtained for the automatic *TREC-5* submission simply due to there being a greater potential for savings when dealing with larger queries. Even though significant improvements were obtained even when using the shorter automatically generated queries our results suggest that the greatest potential for our QSR techniques lie in efficiently dealing with larger queries. It can be argued that in the vast majority of cases users only enter a very small number of query terms when expressing their information need and our QSR techniques will not have a significant impact on retrieval response time. However it is our contention that in order for IR system to provide the quality of results today's users demand some sort of internal 'magical' processing must be carried out by the IR search engine. This internal 'magical' processing usually involves enriching the internal representation of the users initial information need. This enriching process is often carried out by an automatic or manual query expansion process. An example of this is the BORGES information filtering service [Smeaton 1996] in which user profiles are expanded in a semi-automatic fashion in order to improve the effectiveness of the results. In such situations IR systems are required to efficiently deal with short initial user queries which are expanded in some manner in order to improve the quality of results obtained from the system.

The process of expanding an initial user query in an automatic or manual manner has many pitfalls not least of which is the introduction of additional noisy terms which degrade system performance both in terms of efficiency and effectiveness. Our research has shown that QSR has significant beneficial effects on system efficiency and also has the possibility of improving system effectiveness slightly.

8.5 Summary.

This Chapter described our final set of system evaluation experiments in which we extracted from the various TREC-3 and TREC-4 experimental runs, optimal system settings for our QSR techniques. These techniques and their optimal settings

were applied to a new TREC collection with a new set of queries. The performance of the system in terms of effectiveness and efficiency was recorded and analysed. The TREC-5 experiments consisted of 4 experimental runs (two with automatically generated queries and two with manually generated queries) with one automatic and one manual run using our QSR techniques. In all of the TREC-5 experimental runs we achieved significant reductions in the system response time with only minor fluctuations in system effectiveness on a per query level. We then presented our experimental conclusions which are positive in that the techniques developed during the course of our research can be incorporated into inverted file based IR search engines with only minor modifications to the inverted file structure. Our QSR techniques of QTT, PLT and document accumulator restrictions can then be incorporated into the search procedure and used to greatly reduce the amount of data that needs to be processed during retrieval in order to effectively deal with a users query.

8.6 Future Plans.

While what we have done here is to explore the trade-offs between effectiveness and efficiency for collections of documents of *TREC* size our experiments would need to be re-run on other collections of different sizes before we could determine the impact of our thresholding on document collection parameters such as size. From such future experiments we will then try and induce some generalisations as to how our thresholding would perform on different document collections.

The 'InfoLore' system developed during the course of this research is being bundled along with the phrase identification system, the statistical positional information support system, which facilitates DAAG visualisation (described in Chapter 5) and the set of QSR techniques described in this thesis as an IR search engine. At Dublin City University it is and will be used in the following projects.

- Indexing and retrieval on transcriptions of Real Audio recordings of 3rd year databases course lectures (by Alan Smeaton). This is part of the Virtual Lectures project which is ongoing at Dublin City University at present.

- Indexing and retrieval on the 'Dictionary of Computing Hypertext'. The InfoLore system will be used to compute the similarity between nodes in the hypertext with a view to assisting the creation of links between nodes with the hypertext.
- Calculation of a document to document similarity matrix within an Intranet environment. This matrix will be used to perform document clustering with the Intranet based on document content.
- Indexing and retrieval of phoneme representations of radio news articles.
- The InfoLore system will be used as the back-end IR engine to the DAAG visualisation project (starting in the 1st quarter of 1997).
- The InfoLore system is being used for ongoing work on the use of Character Shape Encoding (CSE) in IR (English *TREC-5*) and (French *TREC-6*).
- Plans are also in place to use the InfoLore system in Universidade do Minho in Portugal for investigating database merging.

9. Bibliography.

[Allan *et al* 1993] J. Allan, C. Buckley, G. Salton, *'Automatic routing and ad-hoc retrieval using SMART: TREC 2'*, Proceedings of TREC 2 conference, Gaithersburg, Maryland, USA, November 1994. ---------- 59

[Boyer & Moore 1977] R. Boyer, S. Moore, *'A Fast String Searching Algorithm'*, Communications of the ACM, Vol. 20, Pages 762-772, 1977 ---------- 25

[Brown 1995] E. W. Brown, *'Fast Evaluation of Structured Queries for Information Retrieval'*, Proceedings of SIGIR 95, Seattle, Washington, USA, Pages 30-38, July 1995. ---------- 51

[Brown 1995] E. W. Brown, *'Fast Evaluation of Structured Queries for Information Retrieval'*, SIGIR 95, Seattle, Washington, USA, Pages 30-38, July 1995. --------- 52

[Brown 1996] E. W. Brown, *'Execution Performance Issues in Full-Text Information Retrieval'*, PhD Dissertation, University of Massachusetts Amherst, USA, February 1996. ---------- 51, 52

[Buckley *et al* 1994] C. Buckley, G. Salton, J. Allan, A. Singhal, *'Automatic Query Expansion Using SMART : TREC 3'*, Proceedings of TREC 3, National Institute of Standards and Technology, Washington DC, USA, November 1994.86, 93, 115, 116

[Burkowski 1991] F. J. Burkowski, *'Access Methods for Text Retrieval Systems'*, Tutorial Notes from SIGIR 1991, October 1991. ---------- 25

[Callan 1994] J. Callan, *'Passage level evidence in document retrieval'*, Proceedings of ACM-SIGIR conference, Dublin, Ireland, 1994, Pages 302-309. ---------- passim

[Croft 1995] W. B. Croft, *'What Do People Want from Information Retrieval? (The Top 10 Research Issues for Companies that Use and Sell IR Systems)'*, D-Lib Magazine, November 1995. ---------- 41

[Faloutsos & Jagadish 1992] C. Faloutsos, H. V. Jagadish, *'On b-tree indices for skewed distributions'*, EDBT 1992, pages 363-374, Vancouver, British Columbia, August 1992. ---------- 30

[Frakes & Bazea-Yates 1992] W. B. Frakes, R. Bazea-Yates, *'Information Retrieval, Data Structures & Algorithms'*, Prentice Hall Publishing, 1992. ---------- 25, 91

[Harman & Candela 1990] D. Harman, G. Candela, *'Retrieving Records from a Gigabyte of Text on a Minicomputer using Statistical Ranking'*, Journal of the American Society for Information Science, 41(8), Pages 581-589, 1990. ---------- 57

[Harman 1994] D. Harman, *'Overview of the Third Text REtrieval Conference (TREC-3)'*, National Institute of Standards and Technology, Gaithersburg, MD. 20899, USA, 1994. ---------- 65

[Hearst & Plaunt 1993] Marti A. Hearst, C. Plaunt, *'Subtopic Structuring for Full-Length Document Access'*, Proceedings of SIGIR'93, Pittsburgh, USA, Pages 59-68, June 1993. ---------- 114

[Hearst *et al* 1995] Marti A. Hearst, P. Pirolli, H. Schütze, *'Xerox TREC-4 Site Report'*, Proceedings of TREC-4, Washington DC, USA, November 1995. ------ 104

[Kelledy 1993] F. Kelledy, *'Partitioned Signature Files: Experiments on Large Volumes of Text'*, M.Sc. Dissertation, School of Computer Applications, Dublin City University, Ireland, 1993. ---------- 26

[Knuth *et al* 1977] D. E. Knuth, J. H. Morris, V. R. Pratt, *'Fast pattern matching in strings'*, SIAM J. Comput, 6(2), Pages 323-350, June 1977. ---------- 25

[Lee & Leng 1989] D. L. Lee, C. Leng, *'Partitioned Signature Files: Design Issues and Performance Evaluation'*, ACM Transactions in Office Information Systems, Vol. 7, No. 2, Pages 158-180, April 1989. ---------- 26

[Lee 1987] D. L. Lee, *'A Word-Parallel, Bit-Serial Signature Processor and its Implementation with Magnetic Bubbles'*, Technical Report No. 14, Department of Computer and Information Science, Ohio State University, May 1987. ---------- 26

[Linoff & Stanfill 1993] G. Linoff, C. Stanfill, *'Compression of Indexes with Full Positional Information in Very Large Text Databases'*, Proceedings of ACM SIGIR'81, Pittsburgh, 1993. ---------- 48

[Lucarella 1988] D. Lucarella, *'A Document Retrieval System Based Opon Nearest Neighbour Searching'*, Journal of Information Science, Vol 14, Pages 25-33, 1988.56

[Luhn 1958] H. P. Luhn, *'The automatic creation of literature abstracts'*, IBM Journal of Research and Development, 2, Pages 159-165, 1958. ---------- 19, 118

[Maes 1994] P. Maes, *'Agents the Reduce Work and Information Overload'*, Communications of the ACM, Vol. 37, No. 7, Pages 30-40, 1994. ---------- 36

[Miller 1995] G. A. Miller, *'WordNet: A Lexical Database for English'*, Communications of the ACM, Vol. 38, No. 11, Pages 39-41, 1995.----------39

[Moffat & Zobel 1994] A. Moffat, J. Zobel, *'Fast Ranking in Limited Space'*, Proceedings of the Tenth International Conference on Data Engineering, Houston, USA, Pages 428-437, February 1994. ---------- 125

[Moffat & Zobel 1994] A. Moffat, J. Zobel, *'Information Retrieval Systems for Large Document Collections'*, Proceedings of TREC-4, Washington DC, USA, Pages 85-93, November 1994. ----------53

[Murtagh 1982] F. Murtagh, *'A Very Fast Exact Nearest Neighbour Algorithm for use in Information Retrieval'*, Information Technology, Vol. 1, Pages 275-283, 1982.----------51

[Murtagh 1985] F. Murtagh, *'Multidimensional Clustering Algorithms'*, Physica-Verlag, Würzburg and Vienna, 1985.----------51

[Murtagh 1993] F. Murtagh, *'Search Algorithms for numeric and quantitative data'*, in Information Retrieval: The Case of Astronomy and Related Space Sciences, Kluwer, Dordrecht, Pages 29-48, 1993.----------51

[Persin 1994] M. Persin, *'Document Filtering for Fast Ranking'*, Proceedings of SIGIR 94, Dublin, Ireland, Pages 339-348, July 1994---------- 53, 64

[Persin 1994] M. Persin, *'Document Filtering for Fast Ranking'*, Proceedings of SIGIR 94, Ireland, Pages 339-348, July 1994 ----------31

[Porter 1980] M. F. Porter, *'An Algorithm for Suffix Stripping'*, Program, Vol.14, No. 3, Pages 130-137, 1980.----------22

[Richardson & Smeaton 95] R. Richardson, A. F. Smeaton, *'Automatic word sense disambiguation in a KBIR application'*, Proceedings of the BCSIRSG 17th Colloquium, Pages 299-320, 1995. ----------40

[Salton *et al* 1973] G. Salton, C. S. Yang, *'On the specification of term values in automatic indexing'*, Journal of Documentation, Vol 29, Pages 351-372, 1973. ---24

[Salton *et al* 1983] G. Salton, M. J. McGill, *'Introduction to Modern Information Retrieval'*. McGraw-Hill Publishing, 1983.----------26

[Salton *et al* 1993] G. Salton, J. Allan, C. Buckley, *'Approaches to passage retrieval in full text information systems'*, Proceedings of ACM-SIGIR conference, Pittsburgh, USA, 1993, Pages 49-58.----------59, 62, 64

[Sanderson 94] M. Sanderson, *'Word Sense Disambiguation and Information Retrieval'*, Proceedings of SIGIR'94, Dublin, Pages 142-151, 1994. ----------40

[Schäuble & Mittendorf 1994] P. Schäuble, E. Mittendorf, *'Document and Passage Retrieval Based on Hidden Markov Models'*, Proceedings of ACM-SIGIR conference, Dublin, Ireland, 1994, Pages 318-327. ----------59, 86, 88

[Smeaton & Quigley 1996] A. F. Smeaton, I. Quigley, *'Experiments on Using Semantic Distances Between Words in Image Caption retrieval'*, Proceedings of SIGIR'96, Zurich, Pages 142-151, 1996. ----------41

[Smeaton & van Rijsbergen 1981] A. F. Smeaton, C. J. van Rijsbergen, *'The Nearest Neighbour problem in Information Retrieval'*, Proceedings of ACM SIGIR 81, Oakland, California, USA, Pages 83-87, 1981.----------51, 52, 55

[Smeaton 1995] A. F. Smeaton, ' Tutorial notes from the European Summer School in Information Retrieval ', ESSIR'95, Scotland, September 1995----------38

[Smeaton 1996] A. F. Smeaton, *'Filtering News and WWW Pages with the BORGES Information Filtering Tool'*, Proceedings of the Workshop on Practical Applications of Information Filtering, Basel, Switzerland, 1996. ----------41, 160

[Sparck Jones & van Rijsbergen 1975] K. Sparck Jones, C. J. van Rijsbergen, *'Report on the Need for and Provision of an 'Ideal' Information Retrieval Test Collection'*, British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, United Kingdom, 1975.----------73

[Sparck Jones 1972] K. Sparck Jones, *'A statistical interpretation of term specificity and its application in retrieval'*, Journal of Documentation, Vol. 28, Pages 11-21, 1972.----------23

[van Rijsbergen 1979] C. J. van Rijsbergen, *'Information Retrieval'*, Second Edition, Butterworths & Co. Ltd., 1979.----------passim

[Wilkinson 1994] R. Wilkinson, *'Effective Retrieval of Structured Documents'*, Proceedings of ACM-SIGIR conference, Dublin, Ireland, 1994, Pages 311-317.58, 86

[Witten *et al* 1994] I. H. Witten, A. Moffat, T. C. Bell, *'Managing Gigabytes, Compressing and Indexing Documents and Images'*, Van Nostrand Reinhold, 1994.50

[Zipf 1949] H. P. Zipf, *'Human Behaviour and the Principle of Least Effort'*, Addison- Wesley, Cambridge, Massachusetts, 1949. ---------- 19, 30

[Zobel *et al* 1992] J. Zobel, A. Moffat, R. Sacks-Davis, *'An efficient indexing technique for full-text database systems'*, Proceedings of VLDB, Pages 352-362, August 1992. ---------- 30, 50

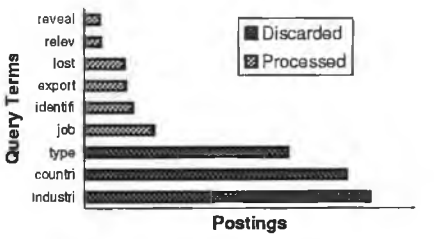
Appendix A

This Appendix contains results on a per query basis for the *TREC-5* manual submission using the optimal settings obtained from the *TREC-3* experimental runs. The results presented for each query (251 to 300) are as follows:

- A tabular description of the Query Space.
- A graphical description of the Query Space.
- A comparison of effectiveness between the query with QSR switched on and the query with QSR switched off.
- A comparison of efficiency between the query with QSR switched on and the query with QSR switched off.

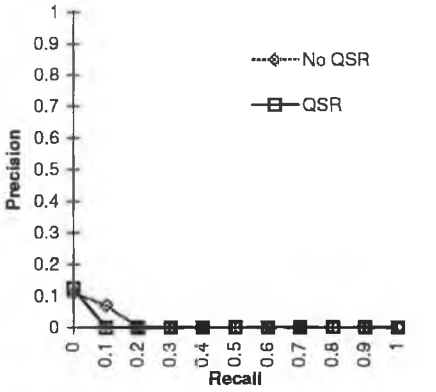
Query: 251

Query Term	NP	QTT	PLT	Processed	Discarded
reveal	14,195	1	14,195	14,195	0
relev	15,178	1	15,162	15,162	16
lost	36,639	1	36,235	36,235	404
export	37,741	1	36,948	36,948	793
identifi	44,281	1	42,908	42,908	1,373
job	63,608	1	61,000	61,000	2,608
type	184,776	0	175,352	0	184,776
countri	237,522	0	223,033	0	237,522
industri	259,266	0	240,858	0	259,266
	893,206			**206,448**	**686,758**



Percentage Reduction: 76.89%

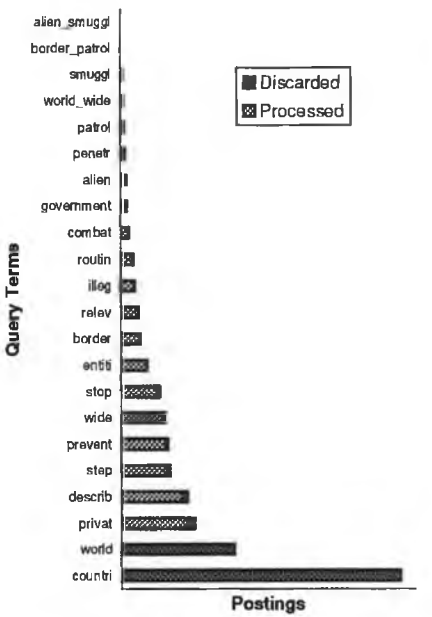
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	579	579
Rel_ret:	**69**	**51**
P. at 0.0	0.1104	0.125
P. at 0.1	0.0723	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0102**	**0.006**
P @ 10 D.	0	0.1
P @ 30 D.	0.0333	0.0333
P @ 100 D	0.08	0.07



	No QSR	QSR	% Red.
Seconds:	14.7	4.9	66.67%
Doc. Acc:	480,613	50,000	89.60%

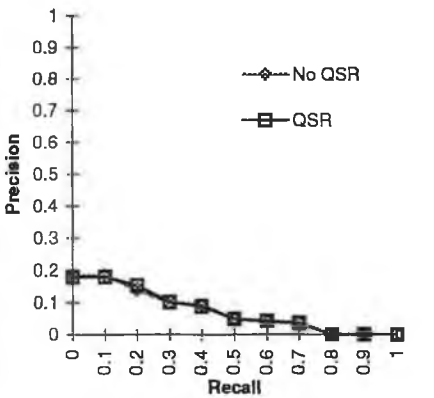
Query: 252

TERM	NP	QTT	PLT	Processed	Discarded
alien_smuggl	46	1	46	46	0
border_patrol	379	1	379	379	0
smuggl	2,051	1	2,051	2,051	0
world_wide	2,935	1	2,923	2,923	12
patrol	3,324	1	3,295	3,295	29
penetr	3,548	1	3,500	3,500	48
alien	4,895	1	4,806	4,806	89
government	5,237	1	5,117	5,117	120
combat	6,830	1	6,641	6,641	189
routin	10,800	1	10,450	10,450	350
illeg	11,958	1	11,513	11,513	445
relev	15,178	1	14,541	14,541	637
border	16,241	1	15,483	15,483	758
entiti	22,570	1	21,409	21,409	1,161
stop	32,448	1	30,624	30,624	1,824
wide	37,694	1	35,396	35,396	2,298
prevent	39,668	1	37,061	37,061	2,607
step	41,546	1	38,617	38,617	2,929
describ	56,667	1	52,403	52,403	4,264
privat	62,828	1	57,801	57,801	5,027
world	96,436	0	88,261	0	96,436
countri	237,522	0	216,258	0	237,522
	710,801			**354,056**	**356,745**



Percentage Reduction: 50.19%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	37	37
Rel_ret:	**28**	**28**
P. at 0.0	0.1842	0.1795
P. at 0.1	0.1842	0.1795
P. at 0.2	0.1404	0.1538
P. at 0.3	0.0992	0.1016
P. at 0.4	0.0938	0.0882
P. at 0.5	0.0461	0.0481
P. at 0.6	0.0404	0.0421
P. at 0.7	0.0356	0.0367
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0651**	**0.0661**
P @ 10 D.	0	0
P @ 30 D.	0.1	0.0667
P @ 100 D	0.11	0.11

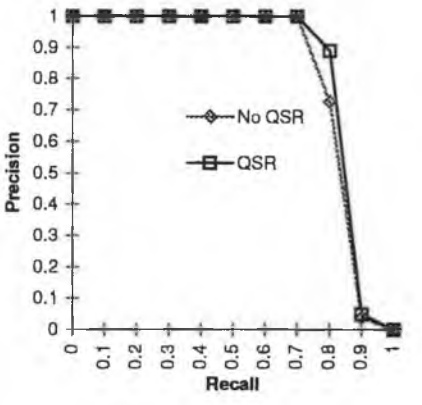


	No QSR	QSR	% Red.
Seconds:	14.6	7.7	47.26%
Doc. Acc:	495,791	50,000	89.92%

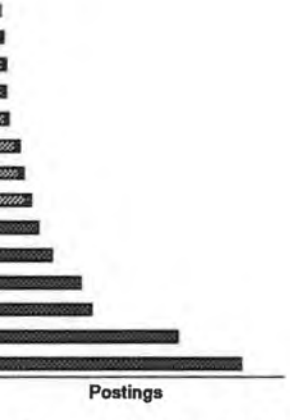
Query: 253

TERM	NP	QTT	PLT	Processed	Discarded
cryonic	8	1	8	8	0
human_bodi	259	1	259	259	0
resuscit	392	1	392	392	0
caus_of_death	547	1	547	547	0
nitrogen	1,545	1	1,543	1,543	2
wealthi	3,115	1	3,103	3,103	12
cure	3,193	1	3,172	3,172	21
viabl	4,108	1	4,069	4,069	39
freez	5,543	1	5,474	5,474	69
suspens	6,892	1	6,787	6,787	105
industri_i	6,963	1	6,837	6,837	126
feasibl	7,679	1	7,518	7,518	161
rush	8,352	1	8,153	8,153	199
quick	8,497	1	8,271	8,271	226
preserv	12,819	1	12,441	12,441	378
storag	14,254	1	13,793	13,793	461
search	16,087	1	15,521	15,521	566
afford	16,607	1	15,975	15,975	632
background	16,815	1	16,127	16,127	688
death	25,116	1	24,018	24,018	1,098
fee	26,660	1	25,418	25,418	1,242
statu	28,029	1	26,643	26,643	1,386
prospect	28,482	1	26,992	26,992	1,490
event	30,910	1	29,205	29,205	1,705
bodi	33,837	1	31,874	31,874	1,963
environ	36,145	1	33,945	33,945	2,200
human	36,323	1	34,008	34,008	2,315
immedi	38,811	1	36,227	36,227	2,584
seek	49,003	1	45,600	45,600	3,403
practic	52,853	1	49,032	49,032	3,821
caus	59,891	1	55,390	55,390	4,501
found	66,708	0	61,504	0	66,708
futur	79,462	0	73,036	0	79,462
determin	107,070	0	98,106	0	107,070
term	117,364	0	107,203	0	117,364
servic	198,999	0	181,202	0	198,999
industri	259,266	0	235,339	0	259,266
	1,408,604			**548,342**	**860,262**

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	10	10
Rel_ret:	9	9
P. at 0.0	1	1
P. at 0.1	1	1
P. at 0.2	1	1
P. at 0.3	1	1
P. at 0.4	1	1
P. at 0.5	1	1
P. at 0.6	1	1
P. at 0.7	1	1
P. at 0.8	0.7273	0.8889
P. at 0.9	0.0398	0.05
P. at 1.0	0	0
Av. P	**0.7767**	**0.7939**
P @ 10 D.	0.7	0.8
P @ 30 D.	0.2667	0.2667
P @ 100 D.	0.08	0.08



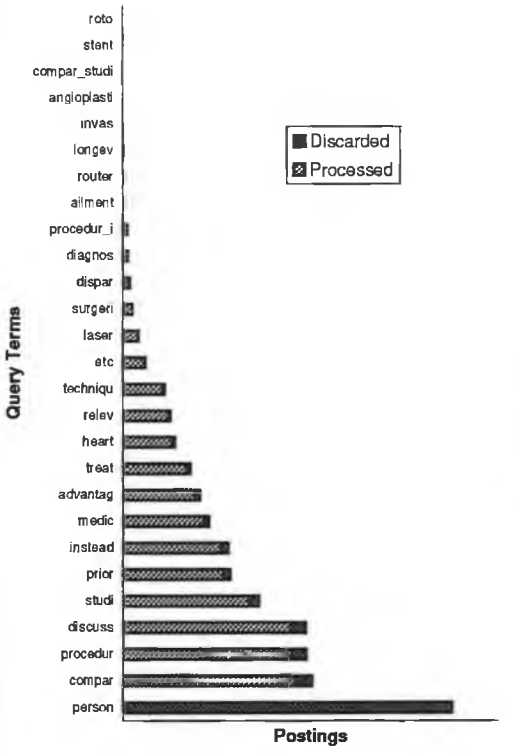
	No QSR	QSR	% Red.
Seconds:	24.8	10.2	58.87%
Doc. Acc:	728,024	50,000	93.13%



Percentage Reduction: 61.07%

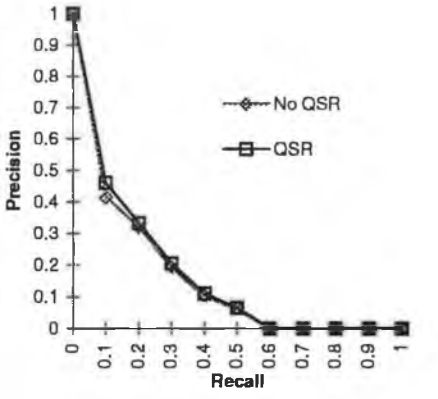
Query: 254

TERM	NP	QTT	PLT	Processed	Discarded
roto	11	1	11	11	0
stent	31	1	31	31	0
compar_studi	77	1	77	77	0
angioplasti	125	1	124	124	1
invas	217	1	215	215	2
longev	385	1	381	381	4
router	514	1	507	507	7
ailment	714	1	702	702	12
procedur_i	1,429	1	1,399	1,399	30
diagnos	1,674	1	1,633	1,633	41
dispar	2,196	1	2,134	2,134	62
surgeri	3,121	1	3,021	3,021	100
laser	4,814	1	4,641	4,641	173
etc	7,173	1	6,888	6,888	285
techniqu	13,355	1	12,774	12,774	581
relev	15,178	1	14,459	14,459	719
heart	16,487	1	15,643	15,643	844
treat	21,437	1	20,257	20,257	1,180
advantag	24,510	1	23,067	23,067	1,443
medic	27,411	1	25,692	25,692	1,719
instead	33,519	1	31,288	31,288	2,231
prior	34,222	1	31,813	31,813	2,409
studi	43,343	1	40,125	40,125	3,218
discuss	58,190	1	53,646	53,646	4,544
procedur	58,527	1	53,732	53,732	4,795
compar	60,366	1	55,188	55,188	5,178
person	105,216	0	95,787	0	105,216
	534,242			**399,448**	**134,794**



Percentage Reduction: 25.23%

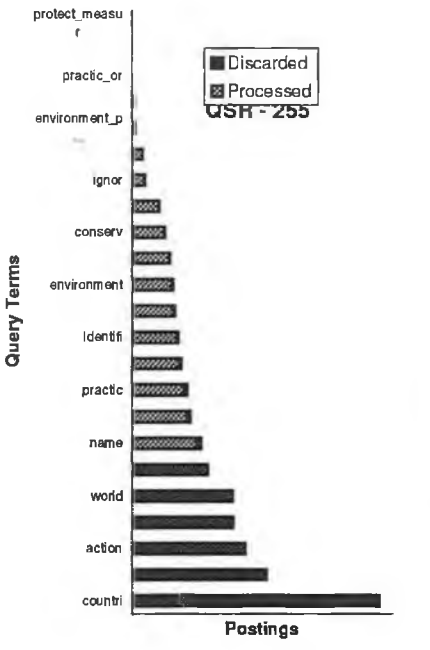
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	85	85
Rel_ret:	**47**	**48**
P. at 0.0	1	1
P. at 0.1	0.4138	0.4615
P. at 0.2	0.3208	0.3333
P. at 0.3	0.1912	0.2031
P. at 0.4	0.1024	0.1093
P. at 0.5	0.0608	0.0644
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1365**	**0.1453**
P @ 10 D.	0.4	0.5
P @ 30 D.	0.4	0.4333
P @ 100 D.	0.2	0.21



	No QSR	QSR	% Red.
Seconds:	13.1	8	38.93%
Doc. Acc:	386,854	50,000	87.08%

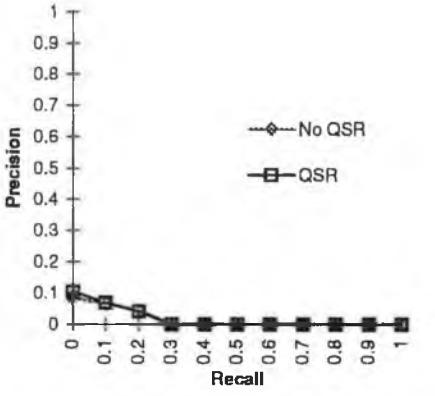
Query: 255

TERM	NP	QTT	PLT	Processed	Discarded
protect_measur	408	1	408	408	0
protect_the_env...	423	1	423	423	0
practic_or	713	1	713	713	0
degrad	2,653	1	2,644	2,644	9
environment_p...	3,008	1	2,984	2,984	24
vital	9,767	1	9,647	9,647	120
ignor	12,007	1	11,805	11,805	202
progress	25,605	1	25,057	25,057	548
conserv	31,229	1	30,419	30,419	810
environ	36,145	1	35,044	35,044	1,101
environment	39,337	1	37,960	37,960	1,377
object	41,181	1	39,552	39,552	1,629
identifi	44,281	1	42,328	42,328	1,953
resourc	47,359	1	45,055	45,055	2,304
practic	52,853	1	50,042	50,042	2,811
measur	56,018	1	52,784	52,784	3,234
name	66,610	1	62,462	62,462	4,148
protect	72,331	0	67,497	0	72,331
world	96,436	0	89,553	0	96,436
control	96,879	0	89,525	0	96,879
action	108,635	0	99,894	0	108,635
intern	129,101	0	118,127	0	129,101
countri	237,522	0	216,252	0	237,522
	1,210,501			**449,327**	**761,174**



Percentage Reduction: 62.88%

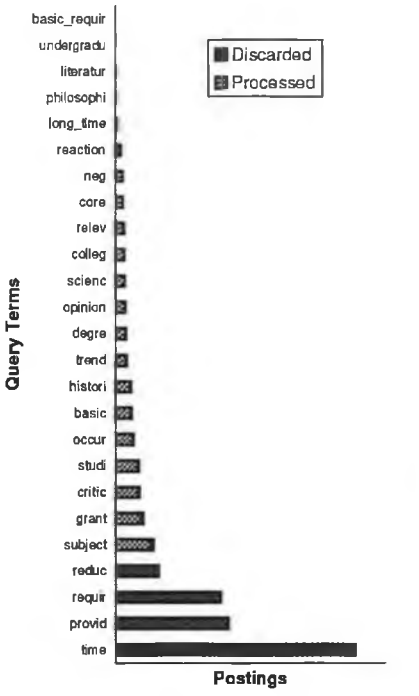
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	109	109
Rel_ret:	**30**	**32**
P. at 0.0	0.0833	0.1053
P. at 0.1	0.068	0.0688
P. at 0.2	0.0421	0.0407
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0152**	**0.0165**
P @ 10 D.	0	0
P @ 30 D.	0.0667	0.0667
P @ 100 D	0.05	0.05



	No QSR	QSR	% Red.
Seconds:	20.6	8	61.17%
Doc. Acc:	662,655	50,000	92.45%

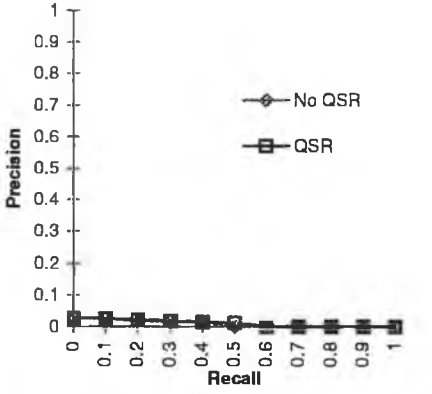
Query: 256

TERM	NP	QTT	PLT	Processed	Discarded
basic_requir	247	1	247	247	0
undergradu	1,148	1	1,148	1,148	0
literatur	2,974	1	2,974	2,974	0
philosophi	3,713	1	3,705	3,705	8
long_time	5,081	1	5,049	5,049	32
reaction	9,940	1	9,836	9,836	104
neg	13,091	1	12,900	12,900	191
core	13,148	1	12,901	12,901	247
relev	15,178	1	14,830	14,830	348
colleg	15,791	1	15,363	15,363	428
scienc	17,082	1	16,548	16,548	534
opinion	17,436	1	16,818	16,818	618
degre	18,803	1	18,058	18,058	745
trend	20,435	1	19,540	19,540	895
histori	28,121	1	26,773	26,773	1,348
basic	29,842	1	28,287	28,287	1,555
occur	32,742	1	30,900	30,900	1,842
studi	43,343	1	40,724	40,724	2,619
critic	44,092	1	41,244	41,244	2,848
grant	51,486	1	47,946	47,946	3,540
subject	69,574	1	64,500	64,500	5,074
reduc	79,858	0	73,702	0	79,858
requir	194,580	0	178,770	0	194,580
provid	208,689	0	190,863	0	208,689
time	443,964	0	404,192	0	443,964
	1,380,358			**430,291**	**950,067**



Percentage Reduction: 68.83%

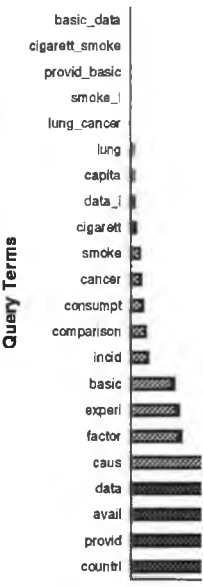
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	22	22
Rel_ret:	**10**	**12**
P. at 0.0	0.0273	0.027
P. at 0.1	0.0273	0.0255
P. at 0.2	0.0177	0.0221
P. at 0.3	0.0172	0.0183
P. at 0.4	0.0133	0.0148
P. at 0.5	0	0.0124
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0083**	**0.0103**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0.01	0.02

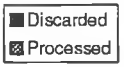


	No QSR	QSR	% Red.
Seconds:	23.6	8.3	64.83%
Doc. Acc:	752,355	50,000	93.35%

Query: 257

TERM	NP	QTT	PLT	Processed	Discarded
basic_data	140	1	140	140	0
cigarett_smoke	232	1	232	232	0
provid_basic	258	1	258	258	0
smoke_i	259	1	258	258	1
lung_cancer	501	1	496	496	5
lung	2,324	1	2,293	2,293	31
capita	2,393	1	2,349	2,349	44
data_i	2,595	1	2,535	2,535	60
cigarett	3,465	1	3,369	3,369	96
smoke	6,145	1	5,946	5,946	199
cancer	6,991	1	6,731	6,731	260
consumpt	8,170	1	7,827	7,827	343
comparison	10,066	1	9,596	9,596	470
incid	11,867	1	11,256	11,256	611
basic	29,842	1	28,165	28,165	1,677
experi	32,994	1	30,982	30,982	2,012
factor	34,803	1	32,515	32,515	2,288
caus	59,891	1	55,670	55,670	4,221
data	84,903	0	78,515	0	84,903
avail	108,167	0	99,513	0	108,167
provid	208,689	0	191,000	0	208,689
countri	237,522	0	216,258	0	237,522
	852,217			**200,618**	**651,599**

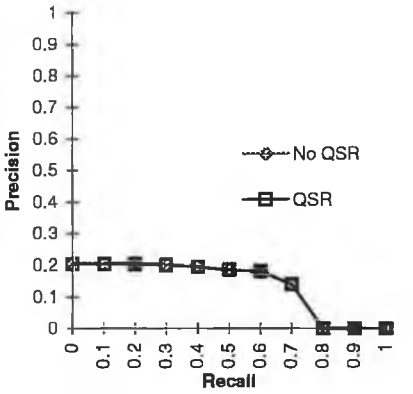




Postings

Percentage Reduction: 76.46%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	135	135
Rel_ret:	**105**	**104**
P. at 0.0	0.2062	0.2048
P. at 0.1	0.2062	0.2048
P. at 0.2	0.2062	0.2048
P. at 0.3	0.202	0.1991
P. at 0.4	0.1964	0.1929
P. at 0.5	0.1831	0.1848
P. at 0.6	0.1796	0.1818
P. at 0.7	0.1401	0.1414
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1309**	**0.1296**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.1333	0.1333
P @ 100 D	0.16	0.15

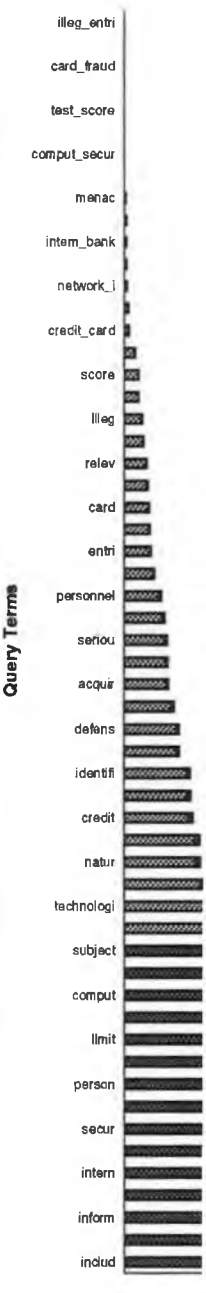


	No QSR	QSR	% Red.
Seconds:	15.7	5.3	66.24%
Doc. Acc:	582,033	50,000	91.41%

Query: 258

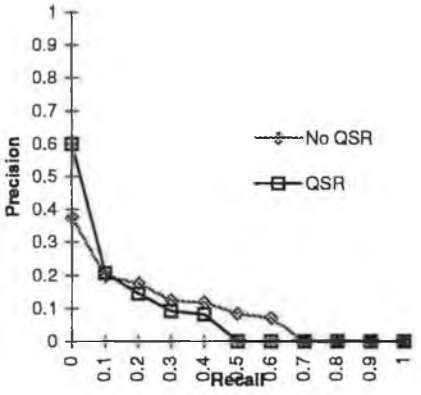
TERM	NP	QTT	PLT	Processed	Discarded
illeg_entri	62	1	62	62	0
credit_card_fraud	68	1	68	68	0
card_fraud	98	1	98	98	0
foreign_agent	116	1	116	116	0
test_score	279	1	279	279	0
defens_or	429	1	429	429	0
comput_secur	491	1	490	490	1
hacker	599	1	597	597	2
menac	679	1	676	676	3
contain_inform	1,453	1	1,443	1,443	10
intern_bank	1,458	1	1,446	1,446	12
comput_network	1,480	1	1,465	1,465	15
network_i	1,720	1	1,699	1,699	21
unauthor	2,439	1	2,405	2,405	34
credit_card	3,142	1	3,092	3,092	50
etc	7,173	1	7,047	7,047	126
score	9,434	1	9,250	9,250	184
fraud	9,623	1	9,417	9,417	206
illeg	11,958	1	11,680	11,680	278
sensit	12,815	1	12,493	12,493	322
relev	15,178	1	14,768	14,768	410
colleg	15,791	1	15,335	15,335	456
card	16,856	1	16,337	16,337	519
instanc	17,111	1	16,552	16,552	559
credit	45,908	1	43,283	43,283	2,625
potenti	50,610	1	47,621	47,621	2,989
natur	51,136	1	48,019	48,019	3,117
test	54,709	1	51,271	51,271	3,438
technologi	58,504	1	54,717	54,717	3,787
contain	69,251	1	64,638	64,638	4,613
subject	69,574	0	64,808	0	69,574
specif	70,952	0	65,958	0	70,952
comput	74,133	0	68,775	0	74,133
foreign	82,835	0	76,692	0	82,835
limit	91,816	0	84,834	0	91,816
consid	95,644	0	88,190	0	95,644
person	105,216	0	96,818	0	105,216
bank	105,852	0	97,204	0	105,852
secur	115,376	0	105,732	0	115,376
author	125,196	0	114,495	0	125,196
intern	129,101	0	117,822	0	129,101
chang	141,578	0	128,942	0	141,578
inform	146,656	0	133,290	0	146,656
countri	237,522	0	215,427	0	237,522
includ	243,806	0	220,667	0	243,806
	2,340,407			**478,908**	**1,861,499**

177

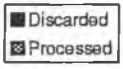


Query Terms

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	115	115
Rel_ret:	**69**	**56**
P. at 0.0	0.375	0.6
P. at 0.1	0.197	0.2063
P. at 0.2	0.1756	0.1453
P. at 0.3	0.1221	0.0923
P. at 0.4	0.1179	0.0819
P. at 0.5	0.0836	0
P. at 0.6	0.0699	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0834**	**0.0715**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.1667	0.1667
P @ 100 D	0.17	0.17



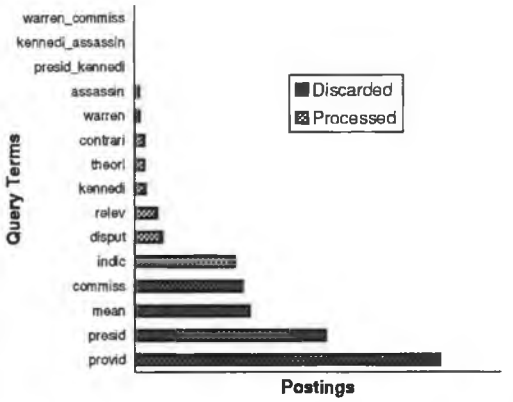
	No QSR	QSR	% Red.
Seconds:	40.3	13.1	67.49%
Doc. Acc:	850,912	50,000	94.12%



Percentage Reduction: 79.54%

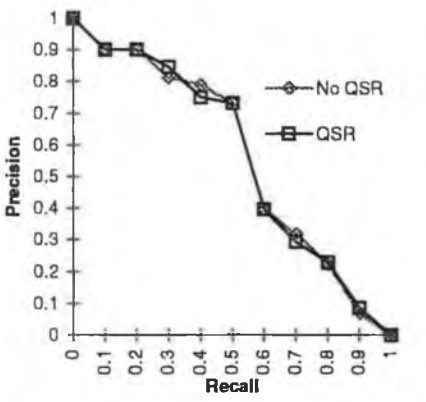
Query: 259

TERM	NP	QTT	PLT	Processed	Discarded
warren_commiss	53	1	53	53	0
kennedi_assassin	154	1	154	154	0
presid_kennedi	399	1	397	397	2
assassin	2,997	1	2,967	2,967	30
warren	3,382	1	3,325	3,325	57
contrari	6,434	1	6,283	6,283	151
theori	6,590	1	6,392	6,392	198
kennedi	7,333	1	7,064	7,064	269
relev	15,178	1	14,520	14,520	658
disput	18,638	1	17,706	17,706	932
indic	68,326	1	64,454	64,454	3,872
commiss	73,351	0	68,705	0	73,351
mean	78,434	0	72,943	0	78,434
presid	130,185	0	120,204	0	130,185
provid	208,689	0	191,298	0	208,689
	620,143			**123,315**	**496,828**



Percentage Reduction: 80.12%

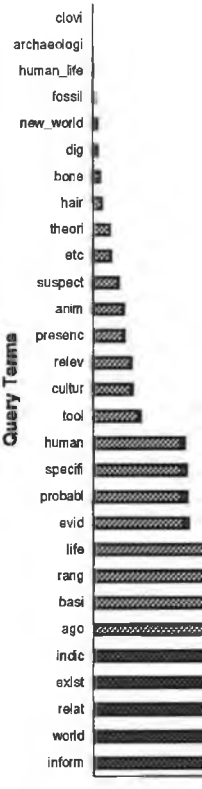
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	36	36
Rel_ret:	**33**	**34**
P. at 0.0	1	1
P. at 0.1	0.9	0.9
P. at 0.2	0.9	0.9
P. at 0.3	0.8125	0.8462
P. at 0.4	0.7895	0.75
P. at 0.5	0.7308	0.7308
P. at 0.6	0.3966	0.3966
P. at 0.7	0.3171	0.2921
P. at 0.8	0.2231	0.2283
P. at 0.9	0.0688	0.0848
P. at 1.0	0	0
Av. P	**0.5392**	**0.5419**
P @ 10 D.	0.9	0.9
P @ 30 D.	0.6333	0.6333
P @ 100 D	0.27	0.27



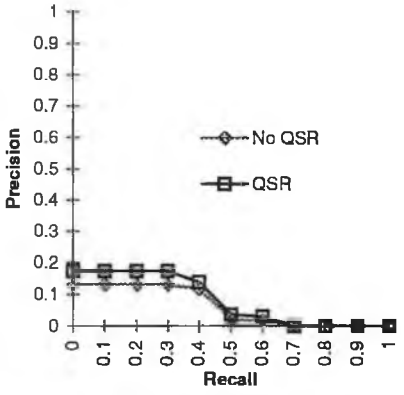
	No QSR	QSR	% Red.
Seconds:	14	4.5	67.86%
Doc. Acc:	461,730	50,000	89.17%

Query: 260

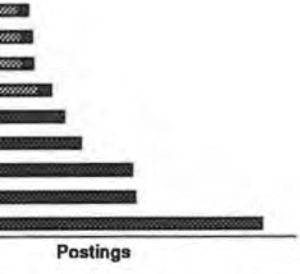
TERM	NP	QTT	PLT	Processed	Discarded
clovi	122	1	122	122	0
archaeologi	221	1	221	221	0
human_life	553	1	553	553	0
fossil	1,043	1	1,042	1,042	1
new_world	1,655	1	1,648	1,648	7
dig	1,792	1	1,778	1,778	14
bone	2,891	1	2,858	2,858	33
hair	3,649	1	3,595	3,595	54
theori	6,590	1	6,469	6,469	121
etc	7,173	1	7,016	7,016	157
suspect	10,173	1	9,915	9,915	258
anim	12,100	1	11,749	11,749	351
presenc	12,342	1	11,940	11,940	402
relev	15,178	1	14,630	14,630	548
cultur	15,639	1	15,019	15,019	620
tool	18,745	1	17,934	17,934	811
human	36,323	1	34,623	34,623	1,700
specifi	37,028	1	35,163	35,163	1,865
probabl	37,173	1	35,168	35,168	2,005
evid	37,806	1	35,632	35,632	2,174
life	53,933	1	50,639	50,639	3,294
rang	55,543	1	51,952	51,952	3,591
basi	56,102	1	52,275	52,275	3,827
ago	63,064	1	58,536	58,536	4,528
indic	68,326	0	63,177	0	68,326
exist	74,900	0	68,988	0	74,900
relat	95,201	0	87,346	0	95,201
world	96,436	0	88,135	0	96,436
inform	146,656	0	133,509	0	146,656
	968,357			**460,477**	**507,880**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	22	22
Rel_ret:	**14**	**15**
P. at 0.0	0.1311	0.1739
P. at 0.1	0.1311	0.1739
P. at 0.2	0.1311	0.1739
P. at 0.3	0.1311	0.1739
P. at 0.4	0.119	0.1389
P. at 0.5	0.0226	0.0374
P. at 0.6	0.0168	0.0302
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0525**	**0.0662**
P @ 10 D.	0	0
P @ 30 D.	0.1	0.1333
P @ 100 D.	0.1	0.1



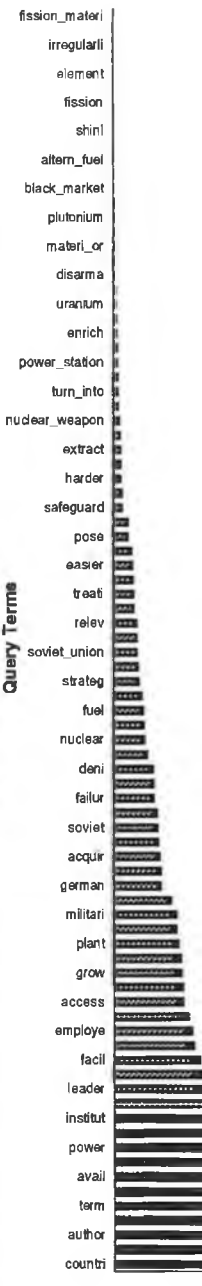
	No QSR	QSR	% Red.
Seconds:	18.6	8.6	53.76%
Doc. Acc:	573,533	50,000	91.28%



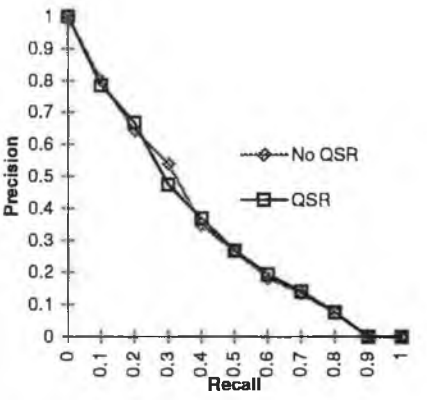
Percentage Reduction: 52.45%

Query: 261

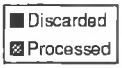
TERM	NP	QTT	PLT	Processed	Discarded
fission_materi	34	1	34	34	0
disgruntl_employe	85	1	85	85	0
irregularli	112	1	112	112	0
underpaid	184	1	184	184	0
element	210	1	210	210	0
author_believ	236	1	236	236	0
fission	242	1	242	242	0
enrich_uranium	260	1	260	260	0
shini	380	1	380	380	0
fuel_l	459	1	458	458	1
altern_fuel	467	1	466	466	1
nuclear_materi	660	1	658	658	2
black_market	785	1	781	781	4
disgruntl	848	1	843	843	5
plutonium	926	1	919	919	7
sphere	978	1	970	970	8
materi_or	1,053	1	1,043	1,043	10
warhead	1,082	1	1,070	1,070	12
disarma	1,187	1	1,173	1,173	14
suscept	1,532	1	1,512	1,512	20
uranium	1,744	1	1,719	1,719	25
shipyard	2,035	1	2,003	2,003	32
enrich	2,097	1	2,062	2,062	35
dismantl	2,561	1	2,515	2,515	46
power_station	2,674	1	2,622	2,622	52
theft	2,886	1	2,827	2,827	59
access	44,800	1	41,479	41,479	3,321
abl	48,296	1	44,655	44,655	3,641
employe	50,937	1	47,033	47,033	3,904
real	51,916	1	47,873	47,873	4,043
facil	55,698	1	51,290	51,290	4,408
union	56,474	1	51,934	51,934	4,540
leader	56,652	1	52,027	52,027	4,625
former	63,768	1	58,483	58,483	5,285
institut	70,263	0	64,352	0	70,263
form	71,419	0	65,321	0	71,419
power	82,890	0	75,709	0	82,890
believ	98,652	0	89,982	0	98,652
avail	108,167	0	98,526	0	108,167
addition	111,989	0	101,867	0	111,989
term	117,364	0	106,610	0	117,364
call	124,865	0	113,268	0	124,865
author	125,196	0	113,411	0	125,196
market	214,777	0	194,292	0	214,777
countri	237,522	0	214,571	0	237,522
	2,112,900			**695,363**	**1,417,537**

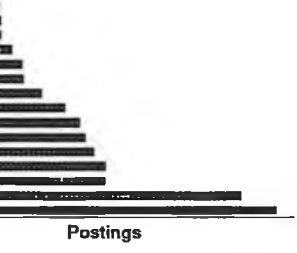


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	87	87
Rel_ret:	**71**	**71**
P. at 0.0	1	1
P. at 0.1	0.8	0.7857
P. at 0.2	0.6429	0.6667
P. at 0.3	0.54	0.4737
P. at 0.4	0.3465	0.3684
P. at 0.5	0.2651	0.2667
P. at 0.6	0.1833	0.193
P. at 0.7	0.1344	0.1415
P. at 0.8	0.0735	0.077
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.3257**	**0.3301**
P @ 10 D.	0.7	0.7
P @ 30 D.	0.6	0.6
P @ 100 D	0.34	0.35



	No QSR	QSR	% Red.
Seconds:	43.9	19.4	55.81%
Doc. Acc:	849,810	50,000	94.12%

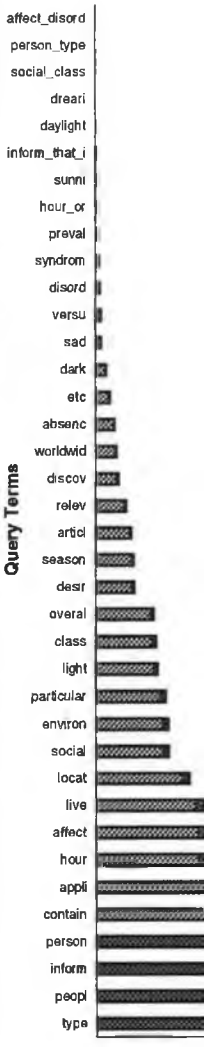




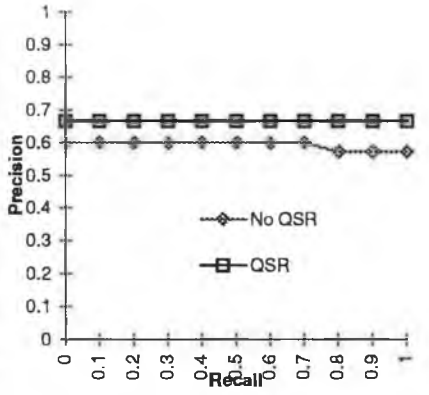
Percentage Reduction: 67.08%

Query: 262

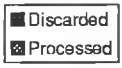
TERM	NP	QTT	PLT	Processed	Discarded
affect_disord	9	1	9	9	0
person_type	52	1	52	52	0
social_class	80	1	80	80	0
dreari	260	1	260	260	0
daylight	736	1	735	735	1
inform_that_i	943	1	939	939	4
sunni	1,194	1	1,186	1,186	8
hour_or	1,232	1	1,221	1,221	11
preval	1,357	1	1,341	1,341	16
syndrom	2,054	1	2,024	2,024	30
disord	2,106	1	2,069	2,069	37
versu	2,887	1	2,829	2,829	58
sad	2,892	1	2,826	2,826	66
dark	5,327	1	5,190	5,190	137
etc	7,173	1	6,969	6,969	204
absenc	9,652	1	9,351	9,351	301
worldwid	10,488	1	10,132	10,132	356
discov	11,337	1	10,921	10,921	416
relev	15,178	1	14,579	14,579	599
articl	17,829	1	17,076	17,076	753
season	18,954	1	18,101	18,101	853
desir	19,288	1	18,366	18,366	922
overal	28,961	1	27,496	27,496	1,465
class	30,148	1	28,540	28,540	1,608
light	30,889	1	29,155	29,155	1,734
particular	34,834	1	32,782	32,782	2,052
environ	36,145	1	33,916	33,916	2,229
social	36,468	1	34,117	34,117	2,351
locat	46,533	1	43,404	43,404	3,129
live	53,510	1	49,764	49,764	3,746
affect	55,078	1	51,069	51,069	4,009
hour	55,715	1	51,505	51,505	4,210
appli	58,074	1	53,524	53,524	4,550
contain	69,251	1	63,633	63,633	5,618
person	105,216	0	96,389	0	105,216
inform	146,656	0	133,945	0	146,656
peopl	147,661	0	134,453	0	147,661
type	184,776	0	167,735	0	184,776
	1,250,943			**625,161**	**625,782**

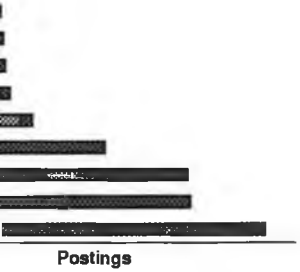


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	4	4
Rel_ret:	4	4
P. at 0.0	0.6	0.6667
P. at 0.1	0.6	0.6667
P. at 0.2	0.6	0.6667
P. at 0.3	0.6	0.6667
P. at 0.4	0.6	0.6667
P. at 0.5	0.6	0.6667
P. at 0.6	0.6	0.6667
P. at 0.7	0.6	0.6667
P. at 0.8	0.5714	0.6667
P. at 0.9	0.5714	0.6667
P. at 1.0	0.5714	0.6667
Av. P	0.5012	0.525
P @ 10 D.	0.4	0.4
P @ 30 D.	0.1333	0.1333
P @ 100 D	0.04	0.04



	No QSR	QSR	% Red.
Seconds:	22.7	11	51.54%
Doc. Acc:	684,119	50,000	92.69%

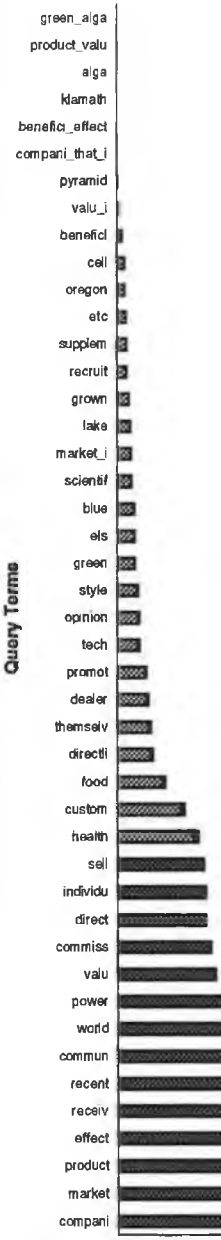




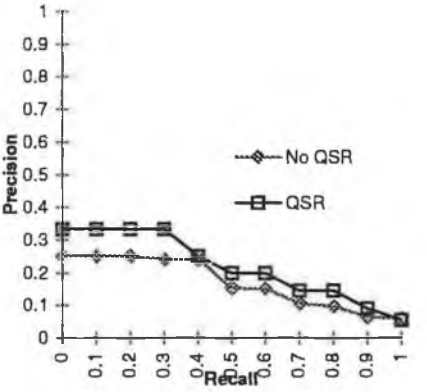
Percentage Reduction: 50.02%

Query: 263

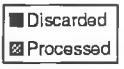
TERM	NP	QTT	PLT	Processed	Discarded
green_alga	19	1	19	19	0
product_valu	166	1	166	166	0
alga	181	1	181	181	0
klamath	236	1	236	236	0
benefici_effect	272	1	272	272	0
compani_that_i	559	1	558	558	1
pyramid	846	1	842	842	4
valu_i	1,951	1	1,939	1,939	12
benefici	3,958	1	3,925	3,925	33
cell	5,854	1	5,791	5,791	63
oregon	5,890	1	5,812	5,812	78
etc	7,173	1	7,061	7,061	112
supplem	7,728	1	7,590	7,590	138
recruit	7,802	1	7,644	7,644	158
grown	9,539	1	9,323	9,323	216
lake	10,712	1	10,444	10,444	268
market_i	11,263	1	10,954	10,954	309
scientif	11,823	1	11,471	11,471	352
blue	13,695	1	13,254	13,254	441
els	13,905	1	13,424	13,424	481
green	14,220	1	13,695	13,695	525
style	16,694	1	16,038	16,038	656
opinion	17,436	1	16,709	16,709	727
tech	17,583	1	16,808	16,808	775
promot	23,351	1	22,266	22,266	1,085
dealer	24,543	1	23,345	23,345	1,198
themselv	26,497	1	25,140	25,140	1,357
directli	28,424	1	26,901	26,901	1,523
food	37,715	1	35,604	35,604	2,111
custom	52,819	1	49,737	49,737	3,082
health	63,917	1	60,036	60,036	3,881
sell	67,782	0	63,505	0	67,782
individu	69,284	0	64,747	0	69,284
direct	69,621	0	64,896	0	69,621
commiss	73,351	0	68,198	0	73,351
valu	77,228	0	71,619	0	77,228
power	82,890	0	76,673	0	82,890
world	96,436	0	88,973	0	96,436
commun	106,141	0	97,674	0	106,141
recent	106,487	0	97,739	0	106,487
receiv	111,050	0	101,663	0	111,050
effect	135,381	0	123,615	0	135,381
product	168,048	0	153,043	0	168,048
market	214,777	0	195,089	0	214,777
compani	270,982	0	245,496	0	270,982
	2,086,229			**417,185**	**1,669,044**

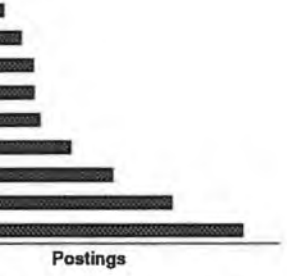


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	15	15
Rel_ret:	**15**	**15**
P. at 0.0	0.25	0.3333
P. at 0.1	0.25	0.3333
P. at 0.2	0.25	0.3333
P. at 0.3	0.24	0.3333
P. at 0.4	0.24	0.25
P. at 0.5	0.1538	0.2
P. at 0.6	0.1515	0.2
P. at 0.7	0.1068	0.1467
P. at 0.8	0.0976	0.1463
P. at 0.9	0.0648	0.0903
P. at 1.0	0.0584	0.0554
Av. P	**0.1607**	**0.1968**
P @ 10 D.	0.1	0.2
P @ 30 D.	0.2	0.2333
P @ 100 D	0.1	0.12



	No QSR	QSR	% Red.
Seconds:	34.1	8.8	74.19%
Doc. Acc:	796,261	50,000	93.72%

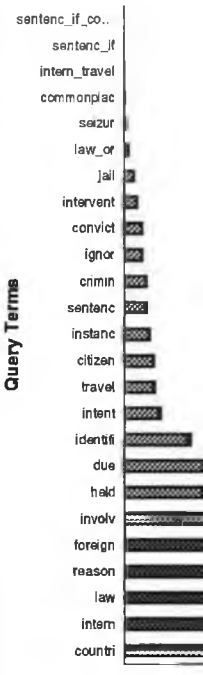




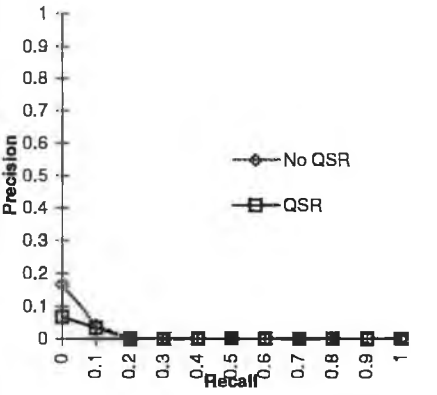
Percentage Reduction: 80.00%

Query: 264

TERM	NP	QTT	PLT	Processed	Discarded
sentenc_if_co...	24	1	24	24	0
sentenc_if	160	1	160	160	0
intern_travel	201	1	201	201	0
commonplac	708	1	706	706	2
seizur	1,535	1	1,525	1,525	10
law_or	3,166	1	3,133	3,133	33
jail	6,608	1	6,511	6,511	97
intervent	8,421	1	8,263	8,263	158
convict	11,805	1	11,534	11,534	271
ignor	12,007	1	11,681	11,681	326
crimin	15,148	1	14,674	14,674	474
sentenc	15,194	1	14,655	14,655	539
instanc	17,111	1	16,433	16,433	678
citizen	19,518	1	18,664	18,664	854
travel	20,351	1	19,375	19,375	976
intent	24,364	1	23,095	23,095	1,269
identifi	44,281	1	41,790	41,790	2,491
due	55,372	1	52,026	52,026	3,346
held	55,850	1	52,243	52,243	3,607
involv	81,131	1	75,553	75,553	5,578
foreign	82,835	0	76,794	0	82,835
reason	86,200	0	79,555	0	86,200
law	89,340	0	82,081	0	89,340
intern	129,101	0	118,073	0	129,101
countri	237,522	0	216,243	0	237,522
	1,017,953			**372,246**	**645,707**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	281	281
Rel_ret:	**39**	**33**
P. at 0.0	0.1667	0.0667
P. at 0.1	0.0412	0.0343
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0068**	**0.0044**
P @ 10 D.	0.1	0
P @ 30 D.	0.0333	0.0333
P @ 100 D	0.05	0.02

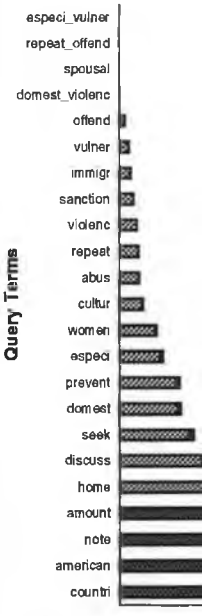


	No QSR	QSR	% Red.
Seconds:	18.6	7.4	60.22%
Doc. Acc:	605,513	50,000	91.74%

Percentage Reduction: 63.43%

Query: 265

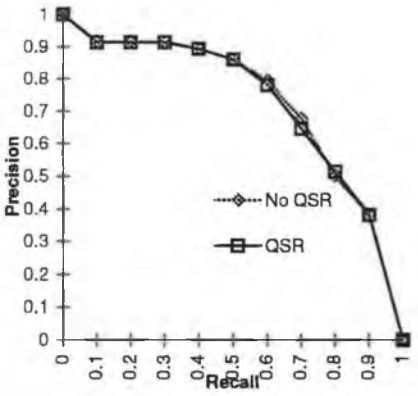
TERM	NP	QTT	PLT	Processed	Discarded
especi_vulner	148	1	148	148	0
repeat_offend	191	1	191	191	0
spousal	202	1	202	202	0
domest_violenc	627	1	625	625	2
offend	3,778	1	3,748	3,748	30
vulner	6,209	1	6,132	6,132	77
immigr	7,458	1	7,332	7,332	126
sanction	9,392	1	9,191	9,191	201
violenc	11,577	1	11,277	11,277	300
repeat	12,878	1	12,485	12,485	393
abus	13,254	1	12,790	12,790	464
cultur	15,639	1	15,020	15,020	619
women	24,696	1	23,607	23,607	1,089
especi	28,553	1	27,164	27,164	1,389
prevent	39,668	1	37,558	37,558	2,110
domest	40,660	1	38,312	38,312	2,348
seek	49,003	1	45,951	45,951	3,052
discuss	58,190	1	54,301	54,301	3,889
home	68,639	1	63,740	63,740	4,899
amount	72,795	0	67,269	0	72,795
note	73,320	0	67,421	0	73,320
american	100,535	0	91,989	0	100,535
countri	237,522	0	216,252	0	237,522
	874,934			**369,774**	**505,160**



Query: 266

TERM	NP	QTT	PLT	Processed	Discarded
scuba_diver	31	1	31	31	0
scuba	127	1	127	127	0
diver	586	1	584	584	2
underwat	611	1	605	605	6
dive	1,326	1	1,306	1,306	20
purpos_i	1,627	1	1,592	1,592	35
apparatu	2,026	1	1,970	1,970	56
breath	4,021	1	3,885	3,885	136
hire	12,779	1	12,267	12,267	512
frequent	13,299	1	12,683	12,683	616
profession	23,037	1	21,827	21,827	1,210
self	25,146	1	23,668	23,668	1,478
job	63,608	1	59,473	59,473	4,135
contain	69,251	0	64,316	0	69,251
purpos	70,718	0	65,237	0	70,718
involv	81.131	0	74,336	0	81.131
	369,324			**140,018**	**229,306**

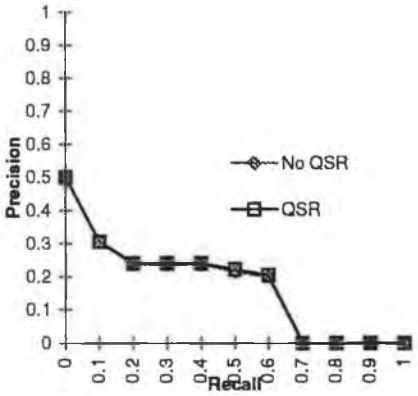
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	147	147
Rel_ret:	**140**	**140**
P. at 0.0	1	1
P. at 0.1	0.9153	0.9138
P. at 0.2	0.9153	0.9138
P. at 0.3	0.9153	0.9138
P. at 0.4	0.8939	0.8939
P. at 0.5	0.8605	0.8605
P. at 0.6	0.7946	0.7797
P. at 0.7	0.6753	0.646
P. at 0.8	0.5021	0.5129
P. at 0.9	0.3789	0.3822
P. at 1.0	0	0
Av. P	**0.7151**	**0.7095**
P @ 10 D.	0.9	0.9
P @ 30 D.	0.8333	0.8333
P @ 100 D.	0.82	0.82



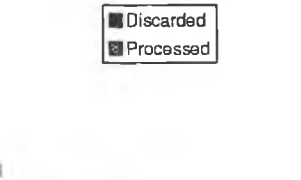
	No QSR	QSR	% Red.
Seconds:	16.8	7.4	55.95%
Doc. Acc:	568,257	50,000	91.20%

Percentage Reduction: 57.74%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	139	139
Rel_ret:	**87**	**87**
P. at 0.0	0.5	0.5
P. at 0.1	0.3043	0.3043
P. at 0.2	0.2381	0.2408
P. at 0.3	0.2381	0.2408
P. at 0.4	0.2381	0.2408
P. at 0.5	0.2169	0.2216
P. at 0.6	0.2	0.2053
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1411**	**0.1413**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.2667	0.2667
P @ 100 D.	0.18	0.18



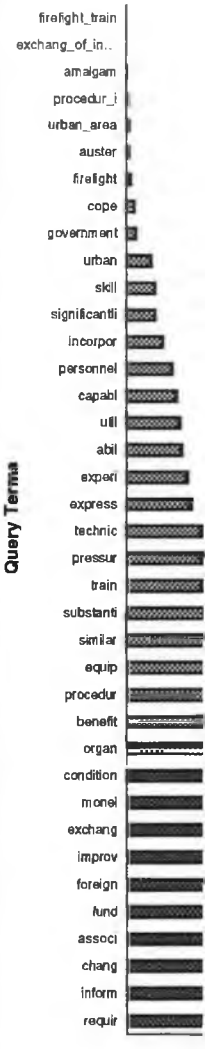
	No QSR	QSR	% Red.
Seconds:	9.8	4.5	54.08%
Doc. Acc:	297,520	50,000	83.19%



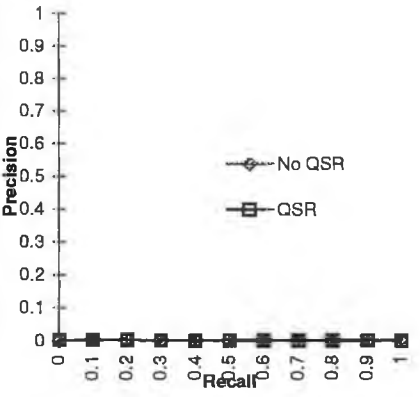
Percentage Reduction: 62.09%

Query: 267

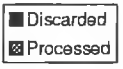
TERM	NP	QTT	PLT	Processed	Discarded
firefight_train	8	1	8	8	0
exchang_of_in...	369	1	369	369	0
amalgam	777	1	777	777	0
procedur_i	1,429	1	1,429	1,429	0
urban_area	1,609	1	1,608	1,608	1
auster	1,812	1	1,805	1,805	7
firefight	2,262	1	2,248	2,248	14
cope	4,312	1	4,273	4,273	39
government	5,237	1	5,175	5,175	62
urban	13,296	1	13,103	13,103	193
skill	15,499	1	15,232	15,232	267
significantli	15,564	1	15,252	15,252	312
incorpor	19,647	1	19,199	19,199	448
personnel	24,729	1	24,097	24,097	632
capabl	27,179	1	26,408	26,408	771
util	29,077	1	28,172	28,172	905
abil	30,058	1	29,039	29,039	1,019
experi	32,994	1	31,784	31,784	1,210
express	35,159	1	33,772	33,772	1,387
technic	41,335	1	39,589	39,589	1,746
pressur	41,912	1	40,025	40,025	1,887
train	42,012	1	40,004	40,004	2,008
substanti	43,593	1	41,389	41,389	2,204
similar	45,918	1	43,469	43,469	2,449
equip	53,183	1	50,198	50,198	2,985
procedur	58,527	1	55,080	55,080	3,447
benefit	65,942	1	61,875	61,875	4,067
organ	67,454	0	63,106	0	67,454
condition	68,845	0	64,217	0	68,845
monei	71,455	0	66,453	0	71,455
exchang	75,608	0	70,105	0	75,608
improv	78,565	0	72,628	0	78,565
foreign	82,835	0	76,346	0	82,835
fund	109,603	0	100,712	0	109,603
associ	115,644	0	105,942	0	115,644
chang	141,578	0	129,307	0	141,578
inform	146,656	0	133,538	0	146,656
requir	194,580	0	176,635	0	194,580
	1,806,262			**625,379**	**1,180,883**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	4	4
Rel_ret:	**1**	**1**
P. at 0.0	0.0027	0.0028
P. at 0.1	0.0027	0.0028
P. at 0.2	0.0027	0.0028
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0007**	**0.0007**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0	0



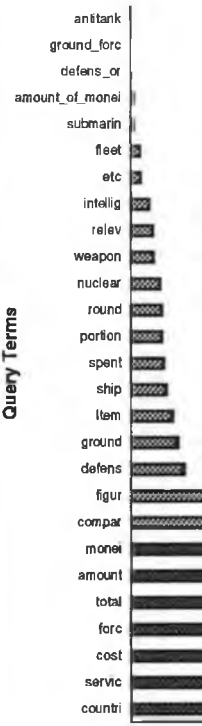
	No QSR	QSR	% Red.
Seconds:	29.8	10.8	63.76%
Doc. Acc:	739,358	50,000	93.24%



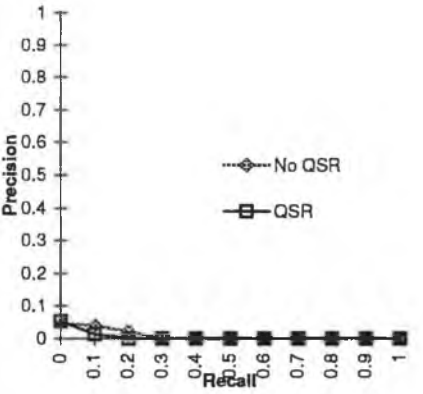
Percentage Reduction: 65.38%

Query: 268

TERM	NP	QTT	PLT	Processed	Discarded
antitank	52	1	52	52	0
ground_forc	318	1	318	318	0
defens_or	429	1	429	429	0
amount_of_monei	2,083	1	2,080	2,080	3
submarin	2,168	1	2,157	2,157	11
fleet	6,732	1	6,672	6,672	60
etc	7,173	1	7,081	7,081	92
intellig	12,671	1	12,461	12,461	210
relev	15,178	1	14,868	14,868	310
weapon	15,768	1	15,385	15,385	383
nuclear	20,049	1	19,486	19,486	563
round	21,360	1	20,678	20,678	682
portion	21,558	1	20,786	20,786	772
spent	22,777	1	21,874	21,874	903
ship	24,573	1	23,505	23,505	1,068
item	29,027	1	27,653	27,653	1,374
ground	32,448	1	30,788	30,788	1,660
defens	36,839	1	34,812	34,812	2,027
figur	56,020	1	52,723	52,723	3,297
compar	60,366	1	56,581	56,581	3,785
monei	71,455	0	66,700	0	71,455
amount	72,795	0	67,671	0	72,795
total	87,331	0	80,848	0	87,331
forc	88,861	0	81,923	0	88,861
cost	142,167	0	130,520	0	142,167
servic	198,999	0	181,931	0	198,999
countri	237,522	0	216,236	0	237,522
	1,286,719			**370,389**	**916,330**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	45	45
Rel_ret:	**13**	**7**
P. at 0.0	0.0476	0.0526
P. at 0.1	0.0394	0.0132
P. at 0.2	0.0213	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0082**	**0.0046**
P @ 10 D.	0	0
P @ 30 D.	0	0.0333
P @ 100 D	0.03	0.04

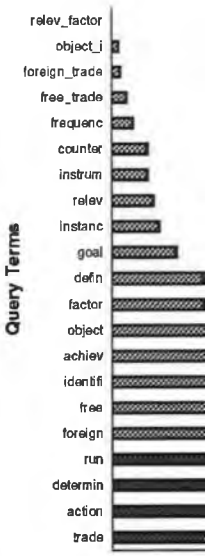


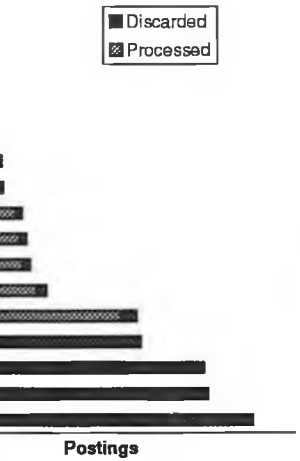
	No QSR	QSR	% Red.
Seconds:	21.7	7.4	65.90%
Doc. Acc:	684,250	50,000	92.69%

Percentage Reduction: 71.21%

Query: 269

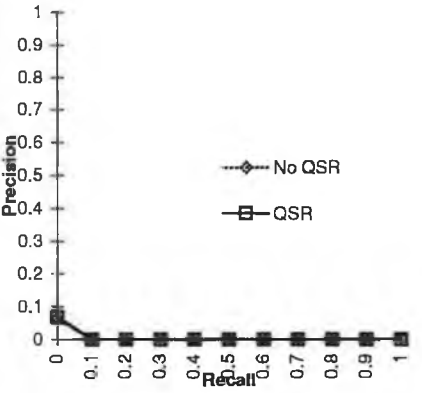
TERM	NP	QTT	PLT	Processed	Discarded
relev_factor	313	1	313	313	0
object_i	2,352	1	2,352	2,352	0
foreign_trade	2,905	1	2,905	2,905	0
free_trade	5,273	1	5,249	5,249	24
frequenc	7,678	1	7,605	7,605	73
instrum	12,873	1	12,686	12,686	187
counter	12,873	1	12,621	12,621	252
relev	15,178	1	14,806	14,806	372
instanc	17,111	1	16,606	16,606	505
goal	23,522	1	22,710	22,710	812
defin	34,166	1	32,816	32,816	1,350
factor	34,803	1	33,254	33,254	1,549
object	41,181	1	39,142	39,142	2,039
achiev	42,981	1	40,638	40,638	2,343
identifi	44,281	1	41,646	41,646	2,635
free	50,085	1	46,854	46,854	3,231
foreign	82,835	1	77,077	77,077	5,758
run	84,196	0	77,923	0	84,196
determin	107,070	0	98,557	0	107,070
action	108,635	0	99,455	0	108,635
trade	124,970	0	113,785	0	124,970
	855,281			**409,280**	**446,001**





Percentage Reduction: 52.15%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	594	594
Rel_ret:	**34**	**27**
P. at 0.0	0.0625	0.0685
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0025**	**0.0014**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0.05	0.05



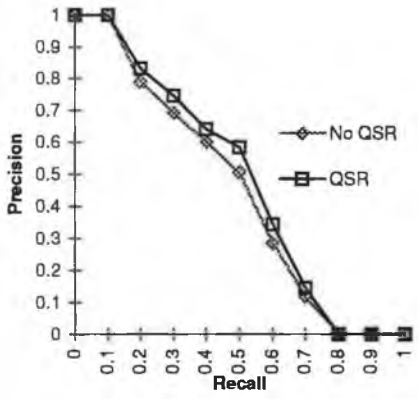
	No QSR	QSR	% Red.
Seconds:	16.3	7.4	54.60%
Doc. Acc:	531,714	50,000	90.60%

Query: 270

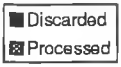
TERM	NP	QTT	PLT	Processed	Discarded
herbal	118	1	118	118	0
amino_acid	126	1	126	126	0
death_and_injuri	215	1	215	215	0
warn_label	245	1	245	245	0
amino	289	1	289	289	0
puriti	615	1	615	615	0
vitamin	699	1	697	697	2
feder_rule	861	1	857	857	4
labell	927	1	921	921	6
dietari	1,139	1	1,129	1,129	10
nearli_half	1,655	1	1,638	1,638	17
exagger	2,618	1	2,586	2,586	32
stringent	3,798	1	3,744	3,744	54
acid	4,062	1	3,995	3,995	67
food_and_drug	4,508	1	4,425	4,425	83
fda	5,503	1	5,390	5,390	113
eventu	17,114	1	16,443	16,443	671
argum	17,570	1	16,845	16,845	725
death	25,116	1	24,027	24,027	1,089
opposit	27,309	1	26,068	26,068	1,241
item	29,027	1	27,648	27,648	1,379
lack	29,337	1	27,882	27,882	1,455
drug	31,471	1	29,844	29,844	1,627
warn	32,382	1	30,641	30,641	1,741
occur	32,742	1	30,913	30,913	1,829
nearli	37,613	1	35,434	35,434	2,179
food	37,715	1	35,452	35,452	2,263
prepar	54,203	1	50,837	50,837	3,366
cover	54,453	1	50,958	50,958	3,495
half	62,102	1	57,987	57,987	4,115
littl	63,268	0	58,944	0	63,268
claim	65,540	0	60,924	0	65,540
address	73,965	0	68,602	0	73,965
control	96,879	0	89,653	0	96,879
american	100,535	0	92,827	0	100,535
recent	106,487	0	98,101	0	106,487
administr	124,500	0	114,436	0	124,500
rule	134,893	0	123,708	0	134,893
feder	163,714	0	149,798	0	163,714
dai	164,554	0	150,224	0	164,554
product	168,048	0	153,063	0	168,048
govern	202,082	0	183,642	0	202,082
includ	243,806	0	221,050	0	243,806
industri	259,266	0	234,527	0	259,266
	2,572,585			**574,609**	**1,997,895**

Query Terms

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	116	116
Rel_ret:	**87**	**87**
P. at 0.0	1	1
P. at 0.1	1	1
P. at 0.2	0.7879	0.8333
P. at 0.3	0.6935	0.7451
P. at 0.4	0.6	0.641
P. at 0.5	0.5043	0.5842
P. at 0.6	0.2846	0.3431
P. at 0.7	0.1199	0.1456
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.4559**	**0.4856**
P @ 10 D.	1	1
P @ 30 D.	0.7667	0.8333
P @ 100 D.	0.52	0.58



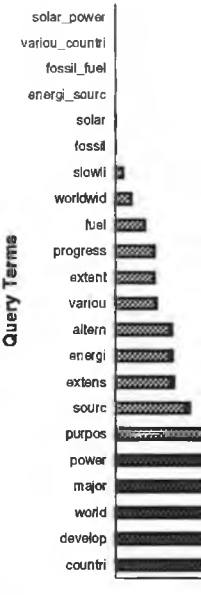
	No QSR	QSR	% Red.
Seconds:	41.7	10.9	73.86%
Doc. Acc:	863,920	50,000	94.21%

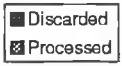


Percentage Reduction: 77.66%

Query: 271

TERM	NP	QTT	PLT	Processed	Discarded
solar_power	123	1	123	123	0
variou_countri	143	1	143	143	0
fossil_fuel	552	1	552	552	0
energi_sourc	581	1	578	578	3
solar	896	1	888	888	8
fossil	1,043	1	1,029	1,029	14
slowli	5,169	1	5,075	5,075	94
worldwid	10,488	1	10,248	10,248	240
fuel	19,450	1	18,912	18,912	538
progress	25,605	1	24,775	24,775	830
extent	25,721	1	24,765	24,765	956
variou	26,999	1	25,867	25,867	1,132
altern	37,184	1	35,448	35,448	1,736
energi	37,411	1	35,487	35,487	1,924
extens	38,320	1	36,166	36,166	2,154
sourc	48,725	1	45,755	45,755	2,970
purpos	70,718	1	66,070	66,070	4,648
power	82,890	0	77,048	0	82,890
major	95,810	0	88,601	0	95,810
world	96,436	0	88,721	0	96,436
develop	155,682	0	142,486	0	155,682
countri	237,522	0	216,258	0	237,522
	1,017,468			**331,881**	**685,587**

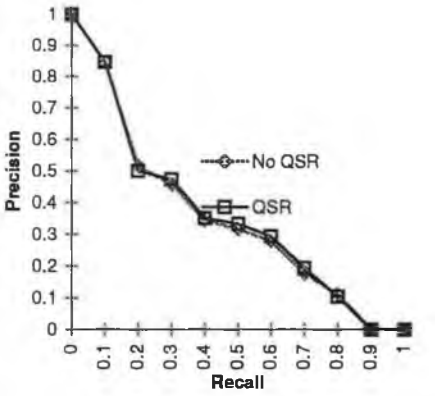




Postings

Percentage Reduction: 67.38%

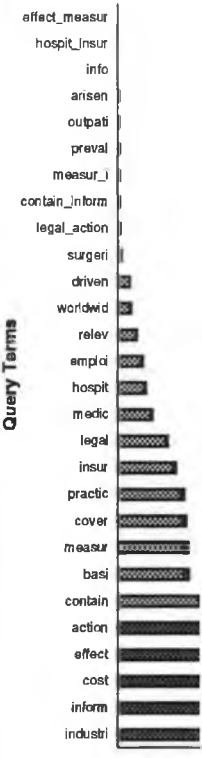
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	86	86
Rel_ret:	**74**	**76**
P. at 0.0	1	1
P. at 0.1	0.8462	0.8462
P. at 0.2	0.5135	0.5
P. at 0.3	0.4576	0.4737
P. at 0.4	0.3455	0.3519
P. at 0.5	0.3162	0.3333
P. at 0.6	0.28	0.2947
P. at 0.7	0.1799	0.1943
P. at 0.8	0.1108	0.1038
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.3471**	**0.3572**
P @ 10 D.	0.8	0.8
P @ 30 D.	0.5333	0.5333
P @ 100 D.	0.33	0.34



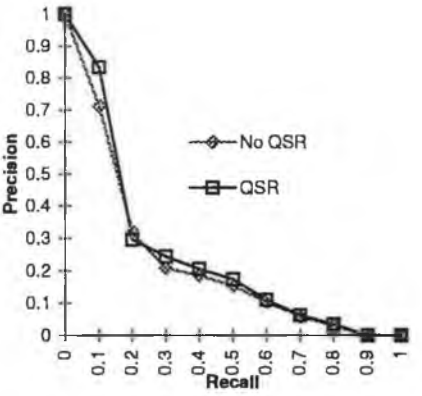
	No QSR	QSR	% Red.
Seconds:	19	6.7	64.74%
Doc. Acc:	619,386	50,000	91.93%

Query: 272

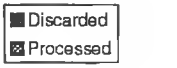
TERM	NP	QTT	PLT	Processed	Discarded
effect_measur	132	1	132	132	0
hospit_insur	287	1	287	287	0
info	611	1	611	611	0
arisen	956	1	955	955	1
outpati	1,111	1	1,106	1,106	5
preval	1,357	1	1,345	1,345	12
measur_i	1,394	1	1,377	1,377	17
contain_inform	1,453	1	1,430	1,430	23
legal_action	2,180	1	2,138	2,138	42
surgeri	3,121	1	3,049	3,049	72
driven	9,730	1	9,470	9,470	260
worldwid	10,488	1	10,169	10,169	319
relev	15,178	1	14,660	14,660	518
emploi	19,671	1	18,927	18,927	744
hospit	22,294	1	21,369	21,369	925
medic	27,411	1	26,172	26,172	1,239
legal	39,598	1	37,662	37,662	1,936
insur	45,993	1	43,574	43,574	2,419
practic	52,853	1	49,877	49,877	2,976
cover	54,453	1	51,185	51,185	3,268
measur	56,018	1	52,449	52,449	3,569
basi	56,102	1	52,320	52,320	3,782
contain	69,251	1	64,326	64,326	4,925
action	108,635	0	100,507	0	108,635
effect	135,381	0	124,751	0	135,381
cost	142,167	0	130,477	0	142,167
inform	146,656	0	134,054	0	146,656
industri	259,266	0	236,028	0	259,266
	1,283,747			**464,590**	**819,157**

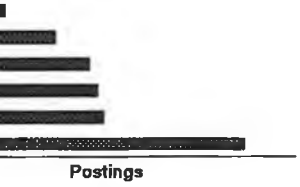


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	36	36
Rel_ret:	**30**	**31**
P. at 0.0	1	1
P. at 0.1	0.7143	0.8333
P. at 0.2	0.3214	0.2963
P. at 0.3	0.2075	0.2444
P. at 0.4	0.186	0.2055
P. at 0.5	0.1538	0.1748
P. at 0.6	0.1038	0.1111
P. at 0.7	0.0584	0.0618
P. at 0.8	0.031	0.0339
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.2238**	**0.2421**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.3	0.2667
P @ 100 D	0.16	0.16



	No QSR	QSR	% Red.
Seconds:	22.3	8.4	62.33%
Doc. Acc:	681,675	50,000	92.67%

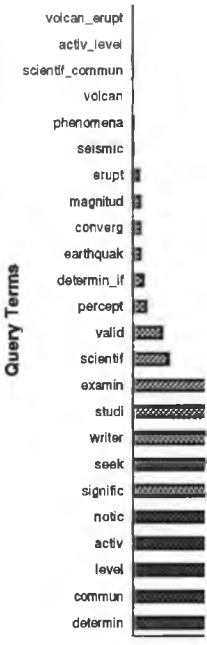




Percentage Reduction: 63.81%

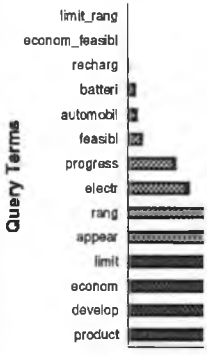
Query: 273

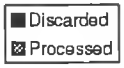
TERM	NP	QTT	PLT	Processed	Discarded
volcan_erupt	120	1	120	120	0
activ_level	159	1	159	159	0
scientif_commun	378	1	378	378	0
volcan	396	1	394	394	2
phenomena	520	1	516	516	4
seismic	688	1	680	680	8
erupt	2,089	1	2,056	2,056	33
magnitud	2,545	1	2,494	2,494	51
converg	2,558	1	2,495	2,495	63
earthquak	2,566	1	2,492	2,492	74
determin_if	3,484	1	3,368	3,368	116
percept	4,407	1	4,242	4,242	165
valid	9,622	1	9,220	9,220	402
scientif	11,823	1	11,278	11,278	545
examin	26,698	1	25,351	25,351	1,347
studi	43,343	1	40,968	40,968	2,375
writer	43,596	1	41,018	41,018	2,578
seek	49,003	1	45,892	45,892	3,111
signific	59,445	1	55,413	55,413	4,032
notic	75,128	0	69,705	0	75,128
activ	95,674	0	88,352	0	95,674
level	102,472	0	94,185	0	102,472
commun	106,141	0	97,095	0	106,141
determin	107,070	0	97,480	0	107,070
	749,925			**248,534**	**501,391**

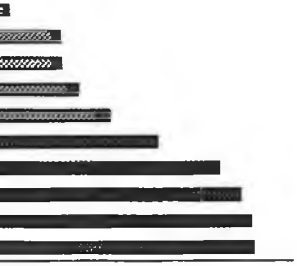


Query: 274

TERM	NP	QTT	PLT	Processed	Discarded
limit_rang	187	1	187	187	0
econom_feasibl	409	1	409	409	0
recharg	639	1	636	636	3
batteri	3,872	1	3,827	3,827	45
automobil	4,874	1	4,783	4,783	91
feasibl	7,679	1	7,481	7,481	198
progress	25,605	1	24,763	24,763	842
electr	32,665	1	31,358	31,358	1,307
rang	55,543	1	52,924	52,924	2,619
appear	66,594	1	62,978	62,978	3,616
limit	91,816	0	86,175	0	91,816
econom	106,546	0	99,239	0	106,546
develop	155,682	0	143,894	0	155,682
product	168,048	0	154,124	0	168,048
	720,159			**189,346**	**530,813**



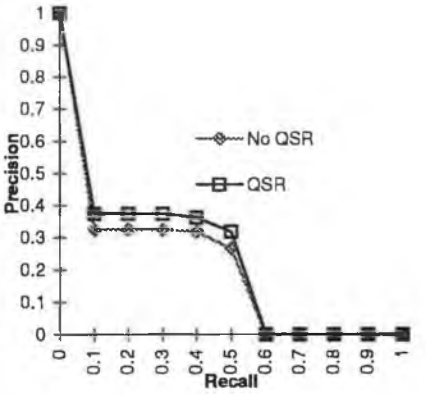




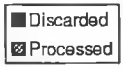
Postings

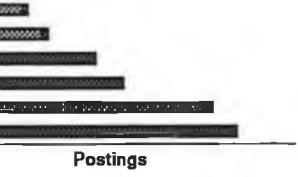
Percentage Reduction: 66.86%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	513	513
Rel_ret:	**261**	**282**
P. at 0.0	1	1
P. at 0.1	0.3248	0.3745
P. at 0.2	0.3248	0.3745
P. at 0.3	0.3248	0.3745
P. at 0.4	0.318	0.362
P. at 0.5	0.2658	0.3185
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1456**	**0.1742**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.2333	0.2667
P @ 100 D	0.27	0.28



	No QSR	QSR	% Red.
Seconds:	16	6.1	61.88%
Doc. Acc:	491,507	50,000	89.83%

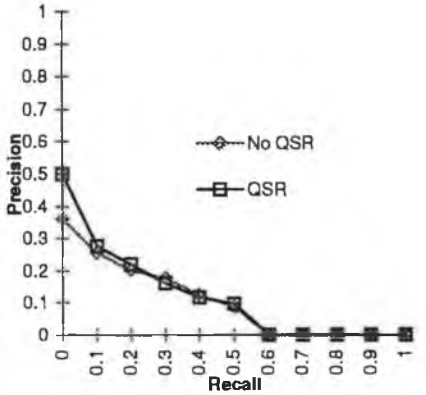




Postings

Percentage Reduction: 73.71%

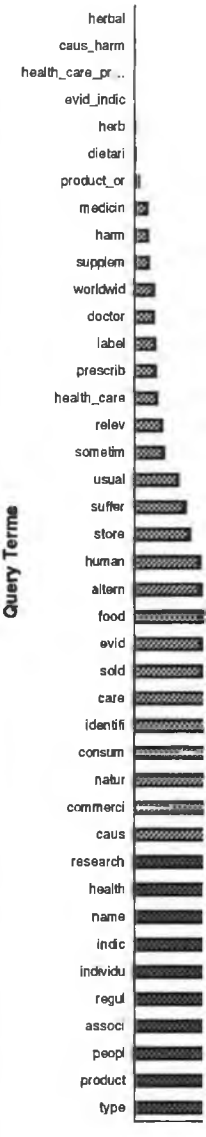
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	119	119
Rel_ret:	**64**	**62**
P. at 0.0	0.3636	0.5
P. at 0.1	0.2568	0.2766
P. at 0.2	0.2026	0.2203
P. at 0.3	0.1756	0.161
P. at 0.4	0.126	0.1161
P. at 0.5	0.0865	0.0972
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0979**	**0.0996**
P @ 10 D.	0.3	0.4
P @ 30 D.	0.2667	0.3
P @ 100 D	0.2	0.2



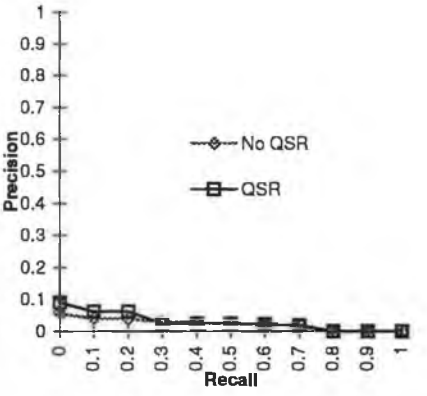
	No QSR	QSR	% Red.
Seconds:	15.4	4.8	68.83%
Doc. Acc:	489,878	50,000	89.79%

Query: 275

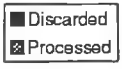
TERM	NP	QTT	PLT	Processed	Discarded
herbal	118	1	118	118	0
caus_harm	143	1	143	143	0
health_care_pr...	161	1	161	161	0
evid_indic	360	1	360	360	0
herb	927	1	927	927	0
dietari	1,139	1	1,136	1,136	3
product_or	2,487	1	2,474	2,474	13
medicin	7,005	1	6,951	6,951	54
harm	7,340	1	7,264	7,264	76
supplem	7,728	1	7,628	7,628	100
worldwid	10,488	1	10,325	10,325	163
doctor	10,596	1	10,403	10,403	193
label	11,383	1	11,146	11,146	237
prescrib	11,708	1	11,433	11,433	275
health_care	12,318	1	11,997	11,997	321
relev	15,178	1	14,742	14,742	436
sometim	16,112	1	15,607	15,607	505
usual	24,156	1	23,335	23,335	821
suffer	28,336	1	27,299	27,299	1,037
store	30,717	1	29,512	29,512	1,205
human	36,323	1	34,803	34,803	1,520
altern	37,184	1	35,530	35,530	1,654
food	37,715	1	35,938	35,938	1,777
evid	37,806	1	35,925	35,925	1,881
sold	39,137	1	37,087	37,087	2,050
care	41,317	1	39,044	39,044	2,273
identifi	44,281	1	41,729	41,729	2,552
consum	48,666	1	45,733	45,733	2,933
natur	51,136	1	47,919	47,919	3,217
commerci	51,581	1	48,201	48,201	3,380
caus	59,891	1	55,808	55,808	4,083
research	60,659	0	56,364	0	60,659
health	63,917	0	59,224	0	63,917
name	66,610	0	61,544	0	66,610
indic	68,326	0	62,949	0	68,326
individu	69,284	0	63,650	0	69,284
regul	104,038	0	95,304	0	104,038
associ	115,644	0	105,631	0	115,644
peopl	147,661	0	134,488	0	147,661
product	168,048	0	152,614	0	168,048
type	184,776	0	167,319	0	184,776
	1,732,400			**650,678**	**1,081,722**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	19	19
Rel_ret:	**14**	**14**
P. at 0.0	0.0556	0.0909
P. at 0.1	0.04	0.0615
P. at 0.2	0.04	0.0615
P. at 0.3	0.0286	0.0235
P. at 0.4	0.0284	0.0235
P. at 0.5	0.0215	0.0235
P. at 0.6	0.0215	0.0229
P. at 0.7	0.0197	0.0185
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.021**	**0.0244**
P @ 10 D.	0	0
P @ 30 D.	0.0333	0.0333
P @ 100 D	0.04	0.04



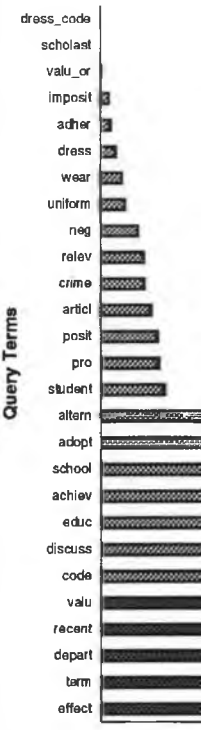
	No QSR	QSR	% Red.
Seconds:	30.7	10.7	65.15%
Doc. Acc:	770,324	50,000	93.51%



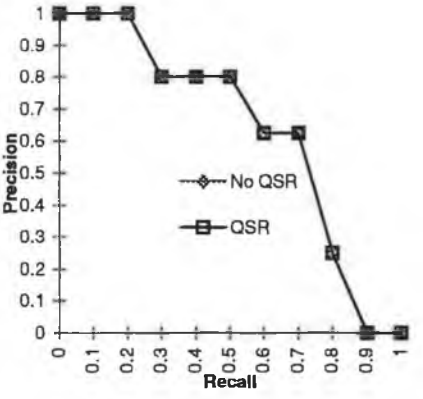
Percentage Reduction: 62.44%

Query: 276

TERM	NP	QTT	PLT	Processed	Discarded
dress_code	121	1	121	121	0
scholast	208	1	208	208	0
valu_or	791	1	791	791	0
imposit	3,022	1	3,018	3,018	4
adher	3,516	1	3,498	3,498	18
dress	5,352	1	5,304	5,304	48
wear	7,564	1	7,467	7,467	97
uniform	8,698	1	8,554	8,554	144
neg	13,091	1	12,824	12,824	267
relev	15,178	1	14,810	14,810	368
crime	15,512	1	15,076	15,076	436
articl	17,829	1	17,259	17,259	570
posit	20,109	1	19,389	19,389	720
pro	20,664	1	19,845	19,845	819
student	22,747	1	21,758	21,758	989
altern	37,184	1	35,424	35,424	1,760
adopt	38,593	1	36,618	36,618	1,975
school	40,281	1	38,065	38,065	2,216
achiev	42,981	1	40,451	40,451	2,530
educ	45,947	1	43,066	43,066	2,881
discuss	58,190	1	54,318	54,318	3,872
code	71,043	1	66,042	66,042	5,001
valu	77,228	0	71,495	0	77,228
recent	106,487	0	98,172	0	106,487
depart	109,295	0	100,341	0	109,295
term	117,364	0	107,297	0	117,364
effect	135,381	0	123,248	0	135,381
	1,034,376			**463,906**	**570,470**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	7	7
Rel_ret:	**6**	**6**
P. at 0.0	1	1
P. at 0.1	1	1
P. at 0.2	1	1
P. at 0.3	0.8	0.8
P. at 0.4	0.8	0.8
P. at 0.5	0.8	0.8
P. at 0.6	0.625	0.625
P. at 0.7	0.625	0.625
P. at 0.8	0.25	0.25
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.6321**	**0.6321**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.2	0.2
P @ 100 D.	0.06	0.06

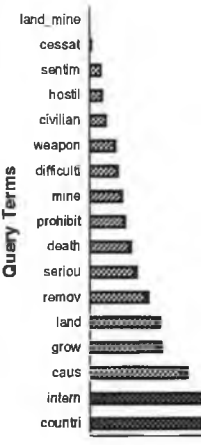


	No QSR	QSR	% Red.
Seconds:	20	8.7	56.50%
Doc. Acc:	590,936	50,000	91.54%

Percentage Reduction: 55.15%

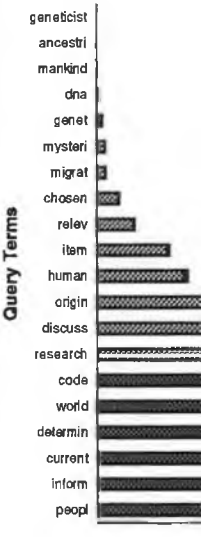
Query: 277

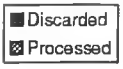
TERM	NP	QTT	PLT	Processed	Discarded
land_mine	154	1	154	154	0
cessat	968	1	968	968	0
sentim	6,875	1	6,862	6,862	13
hostil	7,333	1	7,276	7,276	57
civilian	10,016	1	9,880	9,880	136
weapon	15,768	1	15,461	15,461	307
difficulti	17,263	1	16,826	16,826	437
mine	19,870	1	19,250	19,250	620
prohibit	21,711	1	20,906	20,906	805
death	25,116	1	24,037	24,037	1,079
seriou	28,590	1	27,194	27,194	1,396
remov	35,765	1	33,808	33,808	1,957
land	42,925	1	40,324	40,324	2,601
grow	44,028	1	41,101	41,101	2,927
caus	59,891	1	55,557	55,557	4,334
intern	129,101	0	119,000	0	129,101
countri	237,522	0	217,542	0	237,522
	702,896			**319,604**	**383,292**



Query: 278

TERM	NP	QTT	PLT	Processed	Discarded
geneticist	95	1	95	95	0
ancestri	292	1	292	292	0
mankind	637	1	637	637	0
dna	958	1	952	952	6
genet	2,429	1	2,403	2,403	26
mysteri	3,283	1	3,231	3,231	52
migrat	3,535	1	3,460	3,460	75
chosen	8,947	1	8,711	8,711	236
relev	15,178	1	14,698	14,698	480
item	29,027	1	27,957	27,957	1,070
human	36,323	1	34,793	34,793	1,530
origin	44,577	1	42,465	42,465	2,112
discuss	58,190	1	55,127	55,127	3,063
research	60,659	1	57,147	57,147	3,512
code	71,043	0	66,556	0	71,043
world	96,436	0	89,837	0	96,436
determin	107,070	0	99,180	0	107,070
current	112,595	0	103,705	0	112,595
inform	146,656	0	134,306	0	146,656
peopl	147,661	0	134,449	0	147,661
	945,591			**251,958**	**693,623**

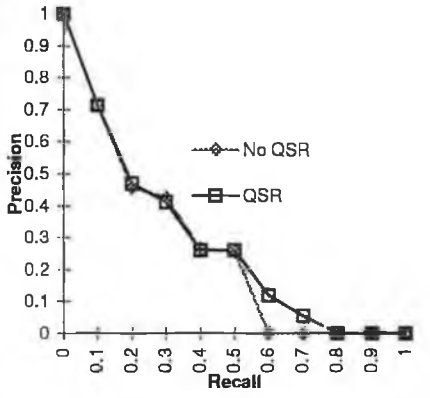




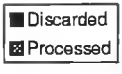
Postings

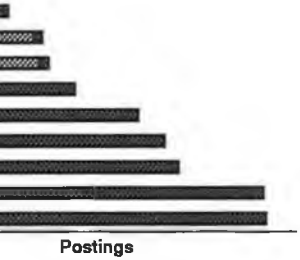
Percentage Reduction: 54.53%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	74	74
Rel_ret:	**44**	**52**
P. at 0.0	1	1
P. at 0.1	0.7143	0.7143
P. at 0.2	0.4571	0.4688
P. at 0.3	0.4259	0.4107
P. at 0.4	0.2636	0.2609
P. at 0.5	0.2569	0.2606
P. at 0.6	0	0.1187
P. at 0.7	0	0.0553
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.2456**	**0.2582**
P @ 10 D.	0.6	0.6
P @ 30 D.	0.4667	0.4667
P @ 100 D	0.26	0.27



	No QSR	QSR	% Red.
Seconds:	14.7	6.4	56.46%
Doc. Acc:	492,323	50,000	89.84%

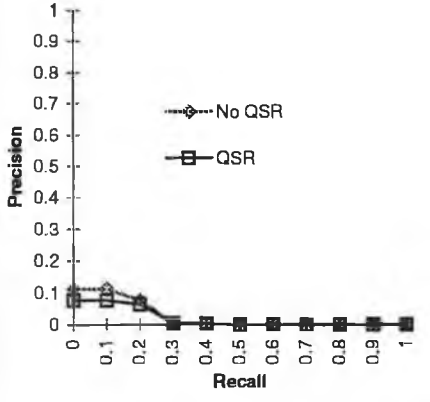




Postings

Percentage Reduction: 73.35%

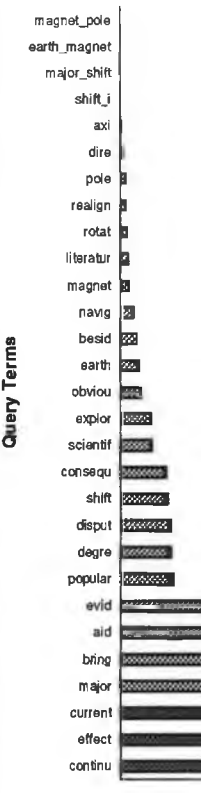
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	7	7
Rel_ret:	**3**	**3**
P. at 0.0	0.1111	0.0769
P. at 0.1	0.1111	0.0769
P. at 0.2	0.0769	0.0645
P. at 0.3	0.0035	0.0039
P. at 0.4	0.0035	0.0039
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0274**	**0.0208**
P @ 10 D.	0.1	0
P @ 30 D.	0.0667	0.0333
P @ 100 D	0.02	0.02



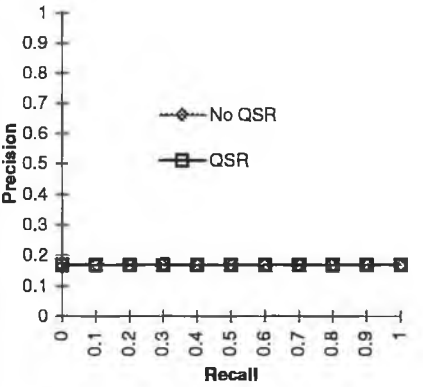
	No QSR	QSR	% Red.
Seconds:	17.2	5.7	66.86%
Doc. Acc:	576,267	50,000	91.32%

Query: 279

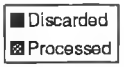
TERM	NP	QTT	PLT	Processed	Discarded
magnet_pole	9	1	9	9	0
earth_magnet	29	1	29	29	0
major_shift	113	1	113	113	0
shift_i	198	1	197	197	1
axi	733	1	730	730	3
dire	1,294	1	1,284	1,284	10
pole	2,169	1	2,144	2,144	25
realign	2,252	1	2,219	2,219	33
rotat	2,555	1	2,508	2,508	47
literatur	2,974	1	2,909	2,909	65
magnet	3,454	1	3,366	3,366	88
navig	4,877	1	4,735	4,735	142
besid	6,305	1	6,100	6,100	205
earth	6,859	1	6,611	6,611	248
obviou	7,903	1	7,589	7,589	314
explor	11,482	1	10,985	10,985	497
scientif	11,823	1	11,269	11,269	554
consequ	17,062	1	16,202	16,202	860
shift	17,400	1	16,461	16,461	939
disput	18,638	1	17,566	17,566	1,072
degre	18,803	1	17,654	17,654	1,149
popular	19,650	1	18,379	18,379	1,271
evid	37,806	1	35,227	35,227	2,579
aid	40,212	1	37,325	37,325	2,887
bring	46,858	1	43,326	43,326	3,532
major	95,810	1	88,247	88,247	7,563
current	112,595	0	103,305	0	112,595
effect	135,381	0	123,728	0	135,381
continu	136,778	0	124,516	0	136,778
	762,022			**353,184**	**408,838**

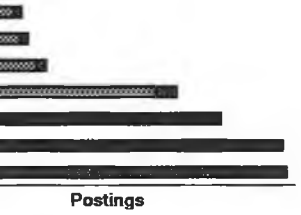


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	2	2
Rel_ret:	**2**	**2**
P. at 0.0	0.1667	0.1667
P. at 0.1	0.1667	0.1667
P. at 0.2	0.1667	0.1667
P. at 0.3	0.1667	0.1667
P. at 0.4	0.1667	0.1667
P. at 0.5	0.1667	0.1667
P. at 0.6	0.1667	0.1667
P. at 0.7	0.1667	0.1667
P. at 0.8	0.1667	0.1667
P. at 0.9	0.1667	0.1667
P. at 1.0	0.1667	0.1667
Av. P	**0.1667**	**0.1667**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.0667	0.0667
P @ 100 D	0.02	0.02



	No QSR	QSR	% Red.
Seconds:	15.5	7.3	52.90%
Doc. Acc:	502,543	50,000	90.05%

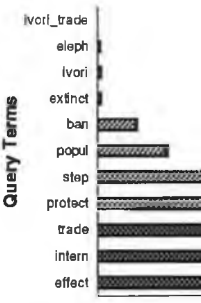


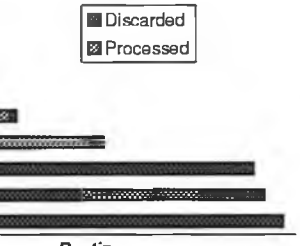


Percentage Reduction: 53.65%

Query: 280

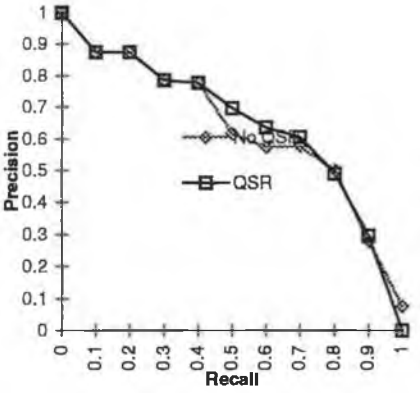
TERM	NP	QTT	PLT	Processed	Discarded
eleph_popul	20	1	20	20	0
ivori_trade	36	1	36	36	0
eleph	941	1	934	934	7
ivori	1,092	1	1,075	1,075	17
extinct	1,178	1	1,150	1,150	28
ban	13,771	1	13,334	13,334	437
popul	24,603	1	23,618	23,618	985
step	41,546	1	39,537	39,537	2,009
protect	72,331	1	68,232	68,232	4,099
trade	124,970	0	116,846	0	124,970
intern	129,101	0	119,633	0	129,101
effect	135,381	0	124,324	0	135,381
	544,970			**147,936**	**397,034**





Percentage Reduction: 72.85%

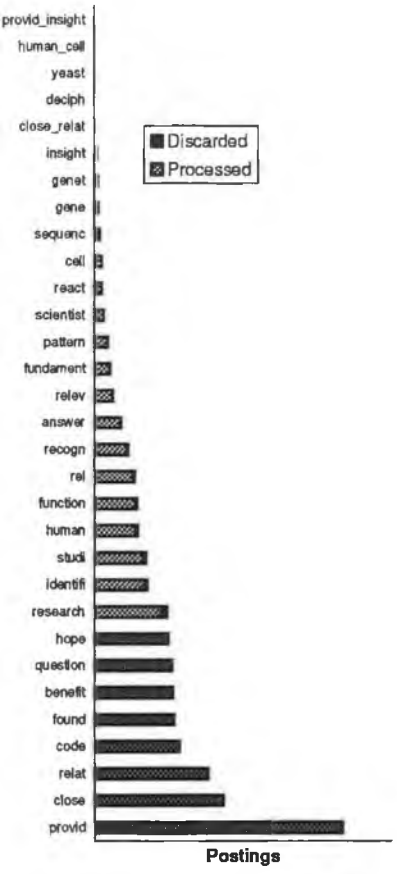
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	32	32
Rel_ret:	**32**	**31**
P. at 0.0	1	1
P. at 0.1	0.875	0.875
P. at 0.2	0.875	0.875
P. at 0.3	0.7857	0.7857
P. at 0.4	0.7778	0.7778
P. at 0.5	0.6154	0.6957
P. at 0.6	0.575	0.6364
P. at 0.7	0.575	0.6053
P. at 0.8	0.5	0.4906
P. at 0.9	0.2788	0.2959
P. at 1.0	0.0748	0
Av. P	**0.614**	**0.6334**
P @ 10 D.	0.7	0.8
P @ 30 D.	0.5667	0.6
P @ 100 D.	0.28	0.29



	No QSR	QSR	% Red.
Seconds:	12.1	4.5	62.81%
Doc. Acc:	415,334	50,000	87.96%

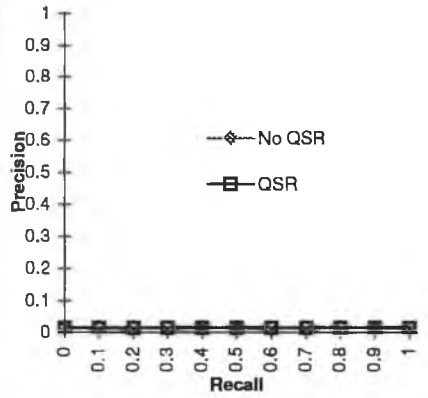
Query: 281

TERM	NP	QTT	PLT	Processed	Discarded
provid_insight	80	1	80	80	0
human_cell	81	1	81	81	0
yeast	203	1	203	203	0
deciph	247	1	247	247	0
close_relat	939	1	936	936	3
insight	2,319	1	2,303	2,303	16
genet	2,429	1	2,404	2,404	25
gene	3,153	1	3,110	3,110	43
sequenc	4,086	1	4,016	4,016	70
cell	5,854	1	5,734	5,734	120
react	6,047	1	5,903	5,903	144
scientist	7,579	1	7,372	7,372	207
pattern	10,744	1	10,414	10,414	330
fundament	12,917	1	12,476	12,476	441
relev	15,178	1	14,607	14,607	571
answer	22,221	1	21,309	21,309	912
recogn	27,972	1	26,727	26,727	1,245
rel	33,263	1	31,668	31,668	1,595
function	35,718	1	33,882	33,882	1,836
human	36,323	1	34,331	34,331	1,992
studi	43,343	1	40,817	40,817	2,526
identifi	44,281	1	41,547	41,547	2,734
research	60,659	1	56,705	56,705	3,954
hope	61,932	0	57,682	0	61,932
question	65,036	0	60,348	0	65,036
benefit	65,942	0	60,962	0	65,942
found	66,708	0	61,440	0	66,708
code	71,043	0	65,188	0	71,043
relat	95,201	0	87,026	0	95,201
close	108,198	0	98,534	0	108,198
provid	208,689	0	189,331	0	208,689
	1,118,385			**356,872**	**761,513**



Percentage Reduction: 68.09%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	1	1
Rel_ret:	**1**	**1**
P. at 0.0	0.0154	0.0149
P. at 0.1	0.0154	0.0149
P. at 0.2	0.0154	0.0149
P. at 0.3	0.0154	0.0149
P. at 0.4	0.0154	0.0149
P. at 0.5	0.0154	0.0149
P. at 0.6	0.0154	0.0149
P. at 0.7	0.0154	0.0149
P. at 0.8	0.0154	0.0149
P. at 0.9	0.0154	0.0149
P. at 1.0	0.0154	0.0149
Av. P	**0.0154**	**0.0149**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0.01	0.01



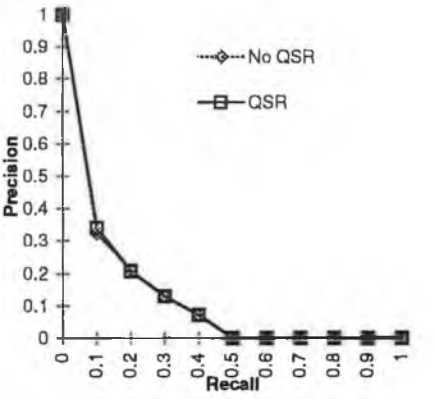
	No QSR	QSR	% Red.
Seconds:	20	7.3	63.50%
Doc. Acc:	613,260	50,000	91.85%

Query: 282

TERM	NP	QTT	PLT	Processed	Discarded
murder_rape	82	1	82	82	0
crime_i	826	1	826	826	0
robberi	1,416	1	1,414	1,414	2
contain_inform	1,453	1	1,443	1,443	10
juvenil	1,935	1	1,911	1,911	24
rape	2,404	1	2,361	2,361	43
violent	6,088	1	5,945	5,945	143
etc	7,173	1	6,965	6,965	208
murder	8,796	1	8,493	8,493	303
relev	15,178	1	14,570	14,570	608
crime	15,512	1	14,805	14,805	707
global	17,528	1	16,632	16,632	896
throughout	25,226	1	23,796	23,796	1,430
occur	32,742	1	30,704	30,704	2,038
arm	34,601	1	32,255	32,255	2,346
contain	69,251	1	64,172	64,172	5,079
world	96,436	0	88,828	0	96,436
inform	146,656	0	134,271	0	146,656
	483,303			**226,374**	**256,929**

Percentage Reduction: 53.16%

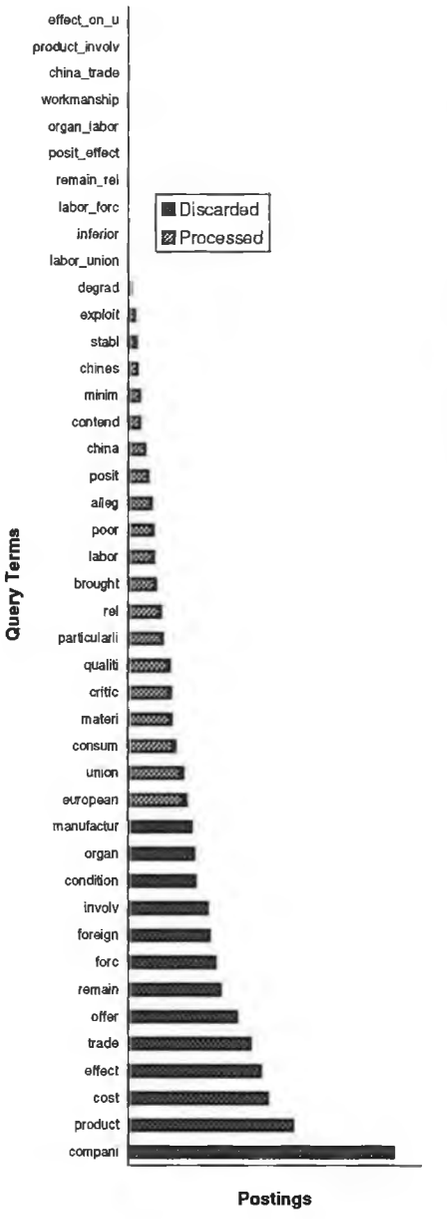
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	131	131
Rel_ret:	**64**	**65**
P. at 0.0	1	1
P. at 0.1	0.3256	0.3415
P. at 0.2	0.2093	0.2077
P. at 0.3	0.1299	0.1303
P. at 0.4	0.0698	0.0713
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1148**	**0.1161**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.4	0.4
P @ 100 D.	0.24	0.24



	No QSR	QSR	% Red.
Seconds:	11.5	5.5	52.17%
Doc. Acc:	368,555	50,000	86.43%

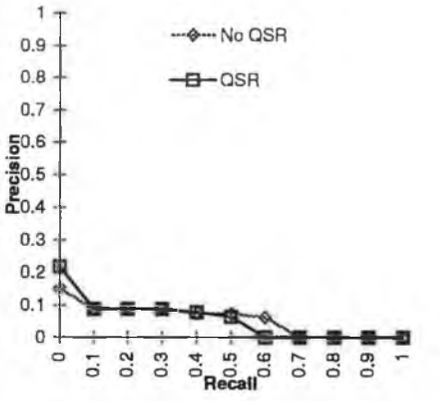
Query: 283

TERM	NP	QTT	PLT	Processed	Discarded
effect_on_u	133	1	133	133	0
product_involv	143	1	143	143	0
china_trade	177	1	177	177	0
workmanship	318	1	318	318	0
organ_labor	385	1	385	385	0
posit_effect	488	1	487	487	1
remain_rel	550	1	547	547	3
labor_forc	585	1	581	581	4
inferior	783	1	775	775	8
labor_union	892	1	881	881	11
degrad	2,653	1	2,615	2,615	38
exploit	6,536	1	6,426	6,426	110
stabl	7,545	1	7,399	7,399	146
chines	9,016	1	8,819	8,819	197
minim	11,701	1	11,417	11,417	284
contend	12,025	1	11,703	11,703	322
china	16,977	1	16,480	16,480	497
posit	20,109	1	19,470	19,470	639
alleg	23,786	1	22,971	22,971	815
poor	25,388	1	24,454	24,454	934
labor	26,277	1	25,245	25,245	1,032
brought	28,034	1	26,863	26,863	1,171
rel	33,263	1	31,791	31,791	1,472
particularli	35,587	1	33,923	33,923	1,664
qualiti	42,379	1	40,291	40,291	2,088
critic	44,092	1	41,810	41,810	2,282
materi	44,766	1	42,337	42,337	2,429
consum	48,666	1	45,904	45,904	2,762
union	56,474	1	53,127	53,127	3,347
european	59,376	1	55,709	55,709	3,667
manufactur	64,464	0	60,322	0	64,464
organ	67,454	0	62,951	0	67,454
condition	68,845	0	64,077	0	68,845
involv	81,131	0	75,309	0	81,131
foreign	82,835	0	76,684	0	82,835
forc	88,861	0	82,040	0	88,861
remain	94,158	0	86,695	0	94,158
offer	111,317	0	102,216	0	111,317
trade	124,970	0	114,441	0	124,970
effect	135,381	0	123,636	0	135,381
cost	142,167	0	129,478	0	142,167
product	168,048	0	152,629	0	168,048
compani	270,982	0	245,441	0	270,982
	2,059,717			**533,181**	**1,526,536**



Percentage Reduction: 74.11%

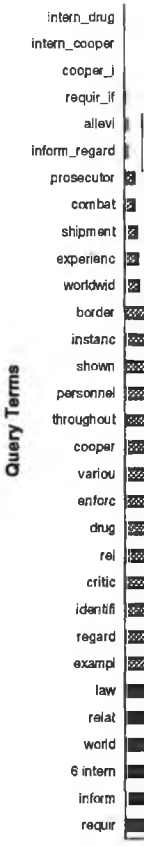
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	84	84
Rel_ret:	**53**	**48**
P. at 0.0	0.15	0.2174
P. at 0.1	0.0881	0.0864
P. at 0.2	0.0881	0.0864
P. at 0.3	0.0881	0.0864
P. at 0.4	0.0727	0.0778
P. at 0.5	0.0727	0.0639
P. at 0.6	0.0617	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0481**	**0.0471**
P @ 10 D.	0	0
P @ 30 D.	0.1333	0.1667
P @ 100 D.	0.06	0.06



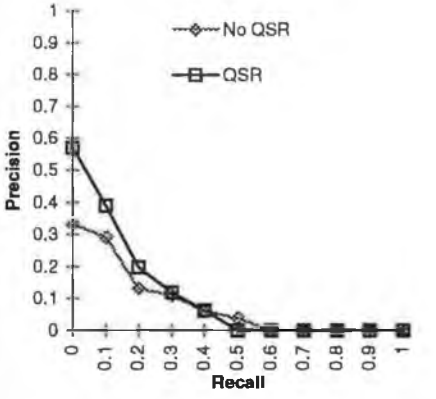
	No QSR	QSR	% Red.
Seconds:	32.5	9.5	70.77%
Doc. Acc:	788,228	50,000	93.66%

Query: 284

TERM	NP	QTT	PLT	Processed	Discarded
intern_drug	208	1	208	208	0
intern_cooper	297	1	297	297	0
cooper_i	333	1	333	333	0
requir_if	1,355	1	1,355	1,355	0
allevi	2,293	1	2,285	2,285	8
inform_regard	2,432	1	2,416	2,416	16
prosecutor	6,705	1	6,637	6,637	68
combat	6,830	1	6,738	6,738	92
shipment	8,646	1	8,499	8,499	147
experienc	9,601	1	9,405	9,405	196
worldwid	10,488	1	10,238	10,238	250
border	16,241	1	15,798	15,798	443
instanc	17,111	1	16,585	16,585	526
shown	19,137	1	18,483	18,483	654
personnel	24,729	1	23,799	23,799	930
throughout	25,226	1	24,190	24,190	1,036
cooper	25,912	1	24,759	24,759	1,153
variou	26,999	1	25,704	25,704	1,295
enforc	28,673	1	27,199	27,199	1,474
drug	31,471	1	29,745	29,745	1,726
rel	33,263	1	31,324	31,324	1,939
critic	44,092	1	41,370	41,370	2,722
identifi	44,281	1	41,395	41,395	2,886
regard	48,209	1	44,900	44,900	3,309
exampl	59,723	1	55,418	55,418	4,305
law	89,340	0	82,593	0	89,340
relat	95,201	0	87,683	0	95,201
world	96,436	0	88,488	0	96,436
6 intern	129,101	0	11,801	0	129,101
inform	146,656	0	13,355	0	146,656
requir	194,580	0	17,653	0	194,580
	1,245,569			**469,080**	**776,489**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	70	70
Rel_ret:	**35**	**32**
P. at 0.0	0.3333	0.5714
P. at 0.1	0.2917	0.3889
P. at 0.2	0.1308	0.1972
P. at 0.3	0.1106	0.1192
P. at 0.4	0.0607	0.0617
P. at 0.5	0.0379	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0743**	**0.1005**
P @ 10 D.	0.3	0.5
P @ 30 D.	0.2333	0.3333
P @ 100 D	0.13	0.15



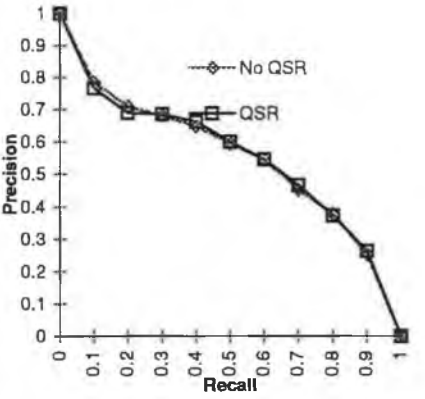
	No QSR	QSR	% Red.
Seconds:	21.5	8.5	60.47%
Doc. Acc:	636,520	50,000	92.14%

Percentage Reduction: 62.34%

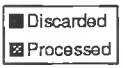
Query: 285

TERM	NP	QTT	PLT	Processed	Discarded
valuabl_inform	164	1	164	164	0
ultim_goal	324	1	324	324	0
mishap	433	1	433	433	0
world_we	860	1	860	860	0
north_sea	1,479	1	1,479	1,479	0
nuclear_power	1,616	1	1,612	1,612	4
shanghai	2,023	1	2,013	2,013	10
submarin	2,168	1	2,152	2,152	16
valuabl	6,116	1	6,056	6,056	60
fleet	6,732	1	6,649	6,649	83
necessarili	7,157	1	7,050	7,050	107
inventori	8,927	1	8,771	8,771	156
navi	9,664	1	9,470	9,470	194
ultim	12,628	1	12,343	12,343	285
intellig	12,671	1	12,352	12,352	319
sea	14,847	1	14,435	14,435	412
relev	15,178	1	14,718	14,718	460
count	16,794	1	16,242	16,242	552
battl	18,610	1	17,951	17,951	659
convent	18,647	1	17,939	17,939	708
tool	18,745	1	17,985	17,985	760
nuclear	20,049	1	19,185	19,185	864
goal	23,522	1	22,448	22,448	1,074
built	23,904	1	22,751	22,751	1,153
french	26,919	1	25,552	25,552	1,367
suffer	28,336	1	26,824	26,824	1,512
entir	29,208	1	27,575	27,575	1,633
construct	43,435	1	40,895	40,895	2,540
north	53,789	1	50,506	50,506	3,283
third	66,428	1	62,203	62,203	4,225
contain	69,251	0	64,669	0	69,251
specif	70,952	0	66,076	0	70,952
power	82,890	0	76,981	0	82,890
forc	88,861	0	82,298	0	88,861
consid	95,644	0	88,335	0	95,644
world	96,436	0	88,820	0	96,436
determin	107,070	0	98,339	0	107,070
current	112,595	0	103,125	0	112,595
inform	146,656	0	133,945	0	146,656
base	166,964	0	152,065	0	166,964
unit	236,217	0	214,533	0	236,217
countri	237,522	0	215,109	0	237,522
	2,002,431			**468,937**	**1,533,494**

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	261	261
Rel_ret:	**236**	**244**
P. at 0.0	1	1
P. at 0.1	0.7838	0.7674
P. at 0.2	0.7105	0.6914
P. at 0.3	0.681	0.687
P. at 0.4	0.6485	0.6627
P. at 0.5	0.5939	0.6009
P. at 0.6	0.5479	0.547
P. at 0.7	0.4534	0.4662
P. at 0.8	0.3793	0.3739
P. at 0.9	0.2527	0.2632
P. at 1.0	0	0
Av. P	**0.5336**	**0.5398**
P @ 10 D.	0.8	0.8
P @ 30 D.	0.7333	0.7667
P @ 100 D.	0.69	0.68



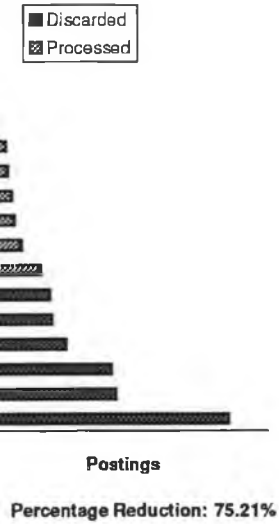
	No QSR	QSR	% Red.
Seconds:	31.2	9.1	70.83%
Doc. Acc:	799,602	50,000	93.75%



Percentage Reduction: 76.58%

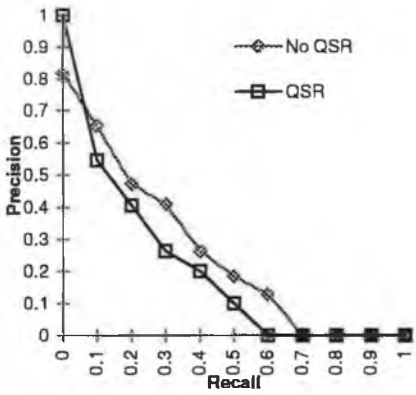
Query: 286

TERM	NP	QTT	PLT	Processed	Discarded
publish_industri	260	1	260	260	0
rise_i	463	1	463	463	0
price_rise	2,501	1	2,498	2,498	3
industri_i	6,963	1	6,916	6,916	47
shortag	7,957	1	7,859	7,859	98
risen	8,894	1	8,735	8,735	159
impos	29,334	1	28,649	28,649	685
paper	30,918	1	30,024	30,024	894
factor	34,803	1	33,604	33,604	1,199
led	37,955	1	36,436	36,436	1,519
materi	44,766	1	42,726	42,726	2,040
publish	65,693	1	62,335	62,335	3,358
rise	74,108	0	69,908	0	74,108
tax	76,398	0	71,644	0	76,398
process	90,712	0	84,563	0	90,712
price	137,644	0	127,550	0	137,644
cost	142,167	0	130,951	0	142,167
industri	259,266	0	237,372	0	259,266
	1,050,802			**260,505**	**790,297**



Percentage Reduction: 75.21%

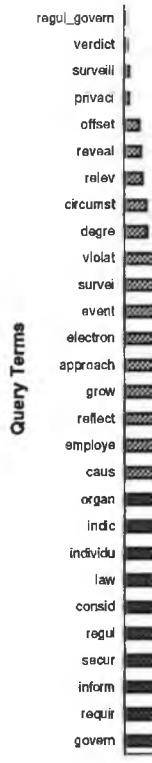
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	142	142
Rel_ret:	**94**	**73**
P. at 0.0	0.8182	1
P. at 0.1	0.6522	0.5455
P. at 0.2	0.4714	0.4054
P. at 0.3	0.4095	0.2638
P. at 0.4	0.2624	0.2007
P. at 0.5	0.1854	0.1007
P. at 0.6	0.1293	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.243**	**0.1788**
P @ 10 D.	0.8	0.6
P @ 30 D.	0.6	0.5333
P @ 100 D	0.41	0.35



	No QSR	QSR	% Red.
Seconds:	18.9	6	68.25%
Doc. Acc:	607,649	50,000	91.77%

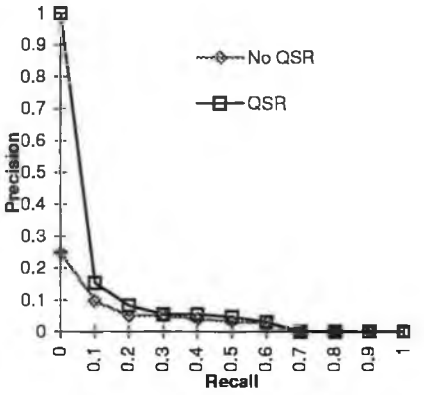
Query: 287

TERM	NP	QTT	PLT	Processed	Discarded
regul_govern	2,172	1	2,172	2,172	0
verdict	2,818	1	2,818	2,818	0
surveill	4,116	1	4,116	4,116	0
privaci	4,323	1	4,319	4,319	4
offset	12,701	1	12,644	12,644	57
reveal	14,195	1	14,079	14,079	116
relev	15,178	1	14,998	14,998	180
circumst	18,469	1	18,181	18,181	288
degre	18,803	1	18,440	18,440	363
violat	24,573	1	24,008	24,008	565
survei	25,451	1	24,772	24,772	679
event	30,910	1	29,971	29,971	939
electron	32,814	1	31,695	31,695	1,119
approach	39,240	1	37,757	37,757	1,483
grow	44,028	1	42,201	42,201	1,827
reflect	46,995	1	44,871	44,871	2,124
employe	50,937	1	48,446	48,446	2,491
caus	59,891	1	56,741	56,741	3,150
organ	67,454	0	63,656	0	67,454
indic	68,326	0	64,226	0	68,326
individu	69,284	0	64,870	0	69,284
law	89,340	0	83,317	0	89,340
consid	95,644	0	88,842	0	95,644
regul	104,038	0	96,254	0	104,038
secur	115,376	0	106,316	0	115,376
inform	146,656	0	134,597	0	146,656
requir	194,580	0	177,860	0	194,580
govern	202,082	0	183,969	0	202,082
	1,600,394			**432,229**	**1,168,165**



Percentage Reduction: 72.99%

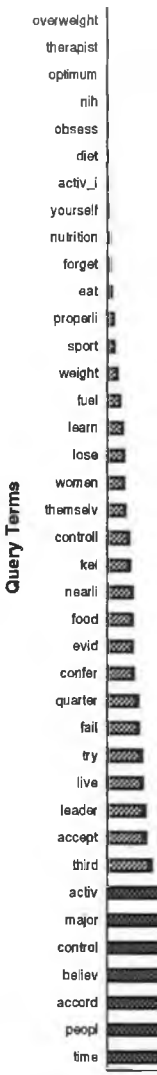
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	40	40
Rel_ret:	**24**	**25**
P. at 0.0	0.25	1
P. at 0.1	0.098	0.1538
P. at 0.2	0.0514	0.0833
P. at 0.3	0.0514	0.0546
P. at 0.4	0.04	0.0546
P. at 0.5	0.0323	0.0461
P. at 0.6	0.0265	0.0294
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0344**	**0.0674**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.0333	0.1333
P @ 100 D.	0.06	0.08



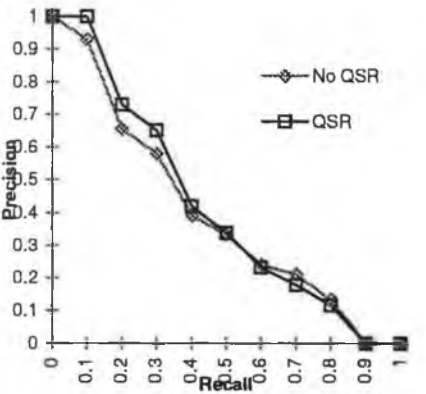
	No QSR	QSR	% Red.
Seconds:	27.6	8.6	68.84%
Doc. Acc:	704,911	50,000	92.91%

Query: 288

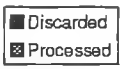
TERM	NP	QTT	PLT	Processed	Discarded
overweight	471	1	471	471	0
therapist	477	1	477	477	0
optimum	943	1	943	943	0
nih	1,737	1	1,737	1,737	0
obsess	1,746	1	1,745	1,745	1
diet	2,360	1	2,352	2,352	8
activ_i	2,580	1	2,565	2,565	15
yourself	3,385	1	3,356	3,356	29
nutrition	3,640	1	3,599	3,599	41
forget	4,440	1	4,378	4,378	62
eat	6,230	1	6,127	6,127	103
properli	9,923	1	9,732	9,732	191
sport	10,910	1	10,671	10,671	239
weight	15,609	1	15,225	15,225	384
fuel	19,450	1	18,919	18,919	531
learn	22,830	1	22,145	22,145	685
lose	24,389	1	23,591	23,591	798
women	24,696	1	23,821	23,821	875
themselv	26,497	1	25,487	25,487	1,010
controll	31,957	1	30,652	30,652	1,305
kei	34,150	1	32,664	32,664	1,486
nearli	37,613	1	35,874	35,874	1,739
food	37,715	1	35,870	35,870	1,845
evid	37,806	1	35,854	35,854	1,952
confer	39,860	1	37,694	37,694	2,166
quarter	46,368	1	43,723	43,723	2,645
fail	47,201	1	44,381	44,381	2,820
try	52,651	1	49,363	49,363	3,288
live	53,510	1	50,024	50,024	3,486
leader	56,652	1	52,808	52,808	3,844
accept	58,356	1	54,239	54,239	4,117
third	66,428	1	61,562	61,562	4,866
activ	95,674	0	88,407	0	95,674
major	95,810	0	88,274	0	95,810
control	96,879	0	88,997	0	96,879
believ	98,652	0	90,359	0	98,652
accord	121,816	0	111,247	0	121,816
peopl	147,661	0	134,451	0	147,661
time	443,964	0	403,047	0	443,964
	1,883,036			**742,049**	**1,140,987**

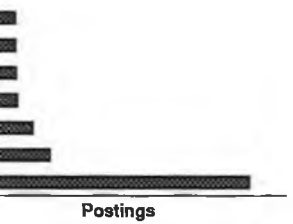


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	92	92
Rel_ret:	**78**	**78**
P. at 0.0	1	1
P. at 0.1	0.9286	1
P. at 0.2	0.6552	0.7308
P. at 0.3	0.58	0.6512
P. at 0.4	0.3918	0.4194
P. at 0.5	0.3333	0.338
P. at 0.6	0.2395	0.2327
P. at 0.7	0.2109	0.1796
P. at 0.8	0.1341	0.1154
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.3984**	**0.4124**
P @ 10 D.	0.9	1
P @ 30 D.	0.6333	0.7
P @ 100 D.	0.38	0.4



	No QSR	QSR	% Red.
Seconds:	31.9	12.6	60.50%
Doc. Acc:	808,639	50,000	93.82%

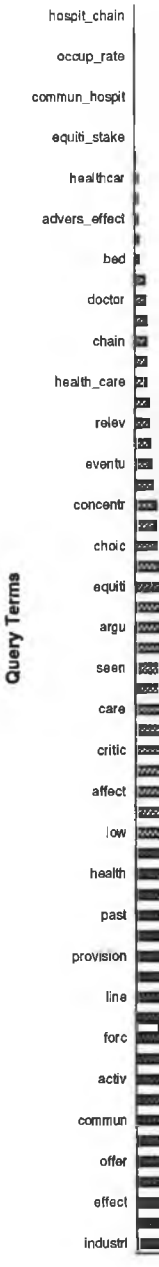




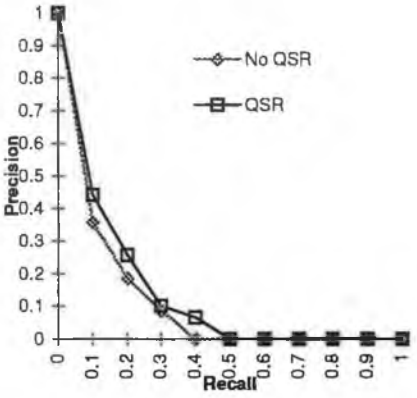
Percentage Reduction: 60.59%

Query: 289

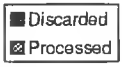
TERM	NP	QTT	PLT	Processed	Discarded
hospit_chain	45	1	45	45	0
profit_hospit	54	1	54	54	0
occup_rate	326	1	326	326	0
rate_drop	329	1	329	329	0
commun_hospit	372	1	372	372	0
hospit_or	596	1	596	596	0
equiti_stake	886	1	886	886	0
bottom_line	1,887	1	1,883	1,883	4
healthcar	3,082	1	3,071	3,071	11
ineffici	3,167	1	3,150	3,150	17
advers_effect	3,246	1	3,223	3,223	23
entrepreneur	4,066	1	4,030	4,030	36
bed	5,313	1	5,257	5,257	56
columbia	10,193	1	10,067	10,067	126
doctor	10,596	1	10,446	10,446	150
bottom	11,824	1	11,636	11,636	188
chain	12,037	1	11,824	11,824	213
occup	12,130	1	11,893	11,893	237
health_care	12,318	1	12,056	12,056	262
advers	14,584	1	14,248	14,248	336
relev	15,178	1	14,801	14,801	377
consum	48,666	1	45,980	45,980	2,686
affect	55,078	1	51,940	51,940	3,138
discuss	58,190	1	54,771	54,771	3,419
low	62,274	1	58,504	58,504	3,770
creat	62,828	0	58,912	0	62,828
health	63,917	0	59,819	0	63,917
corpor	67,095	0	62,673	0	67,095
past	70,364	0	65,601	0	70,364
form	71,419	0	66,457	0	71,419
provision	78,957	0	73,331	0	78,957
profit	82,449	0	76,427	0	82,449
line	82,559	0	76,381	0	82,559
power	82,890	0	76,540	0	82,890
forc	88,861	0	81,894	0	88,861
limit	91,816	0	84,454	0	91,816
activ	95,674	0	87,832	0	95,674
level	102,472	0	93,889	0	102,472
commun	106,141	0	97,062	0	106,141
close	108,198	0	98,749	0	108,198
offer	111,317	0	101,397	0	111,317
rate	118,907	0	108,099	0	118,907
effect	135,381	0	122,834	0	135,381
cost	142,167	0	128,737	0	142,167
industri	259,266	0	234,311	0	259,266
	2,842,887			**785,016**	**2,057,871**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	141	141
Rel_ret:	55	58
P. at 0.0	1	1
P. at 0.1	0.3571	0.4412
P. at 0.2	0.1859	0.2566
P. at 0.3	0.0869	0.1005
P. at 0.4	0	0.0648
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.0977	0.1256
P @ 10 D.	0.5	0.6
P @ 30 D.	0.3667	0.4333
P @ 100 D.	0.24	0.26



	No QSR	QSR	% Red.
Seconds:	44.4	13.1	70.50%
Doc. Acc:	855,010	50,000	94.15%

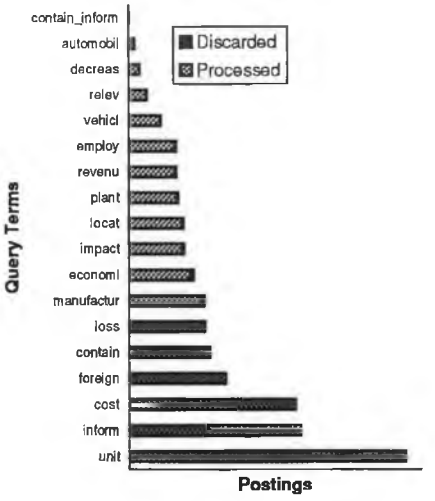


Postings

Percentage Reduction: 72.39%

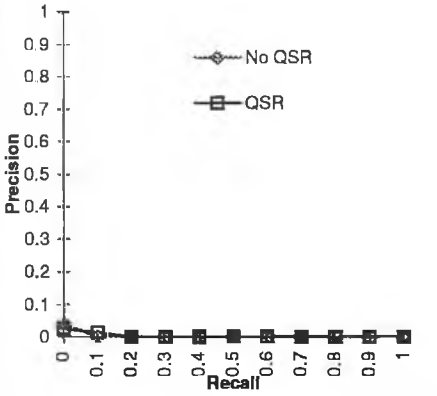
Query: 290

TERM	NP	QTT	PLT	Processed	Discarded
contain_inform	1,453	1	1,453	1,453	0
automobil	4,874	1	4,874	4,874	0
decreas	9,313	1	9,302	9,302	11
relev	15,178	1	15,076	15,076	102
vehicl	27,293	1	26,959	26,959	334
employ	40,336	1	39,618	39,618	718
revenu	40,527	1	39,581	39,581	946
plant	41,960	1	40,747	40,747	1,213
locat	46,533	1	44,930	44,930	1,603
impact	46,830	1	44,956	44,956	1,874
economi	55,130	1	52,618	52,618	2,512
manufactur	64,464	1	61,169	61,169	3,295
loss	65,134	0	61,443	0	65,134
contain	69,251	0	64,942	0	69,251
foreign	82,835	0	77,220	0	82,835
cost	142,167	0	131,741	0	142,167
inform	146,656	0	135,086	0	146,656
unit	236,217	0	216,269	0	236,217
	1,136,151			**381,283**	**754,868**



Percentage Reduction: 66.44%

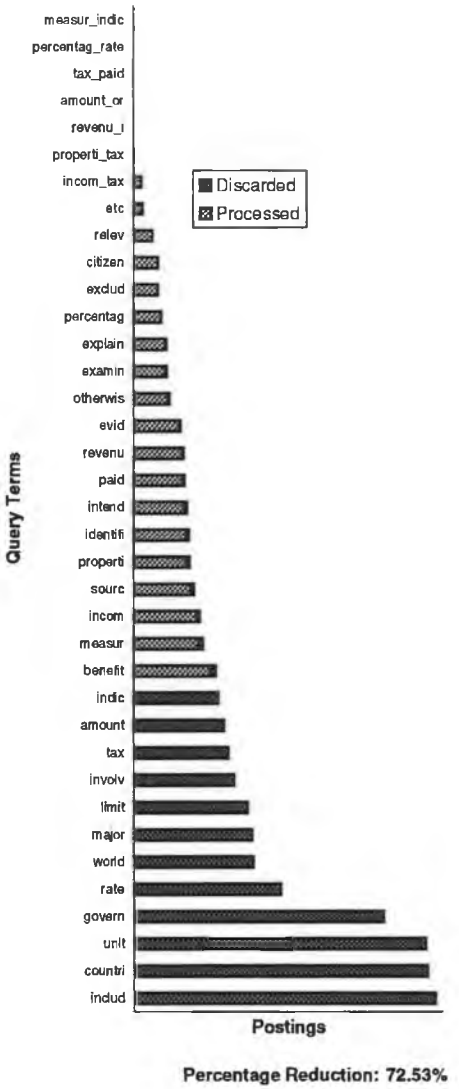
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	119	119
Rel_ret:	**10**	**13**
P. at 0.0	0.04	0.0256
P. at 0.1	0	0.0142
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0013**	**0.0017**
P @ 10 D.	0	0
P @ 30 D.	0.0333	0
P @ 100 D	0.01	0.01



	No QSR	QSR	% Red.
Seconds:	19.7	7.1	63.96%
Doc. Acc:	645,020	50,000	92.25%

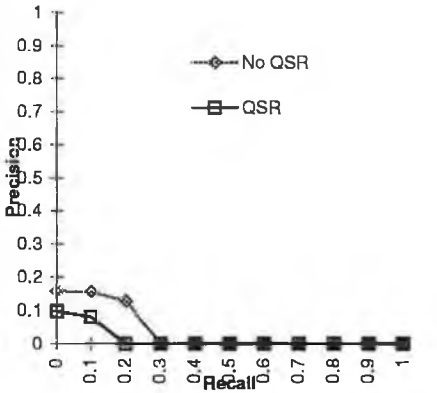
Query: 291

TERM	NP	QTT	PLT	Processed	Discarded
measur_indic	56	1	56	56	0
percentag_rate	300	1	300	300	0
tax_paid	509	1	509	509	0
amount_or	592	1	592	592	0
revenu_i	623	1	622	622	1
properti_tax	809	1	805	805	4
incom_tax	6,024	1	5,984	5,984	40
etc	7,173	1	7,105	7,105	88
relev	15,178	1	14,991	14,991	187
citizen	19,518	1	19,222	19,222	296
exclud	19,609	1	19,256	19,256	353
percentag	22,440	1	21,971	21,971	469
explain	26,304	1	25,680	25,680	624
examin	26,698	1	25,988	25,988	710
otherwis	28,816	1	27,967	27,967	849
evid	37,806	1	36,585	36,585	1,221
revenu	40,527	1	39,102	39,102	1,425
paid	40,681	1	39,135	39,135	1,546
intend	42,811	1	41,061	41,061	1,750
identifi	44,281	1	42,345	42,345	1,936
properti	44,726	1	42,643	42,643	2,083
sourc	48,725	1	46,316	46,316	2,409
incom	53,377	1	50,586	50,586	2,791
measur	56,018	1	52,929	52,929	3,089
benefit	65,942	1	62,117	62,117	3,825
indic	68,326	0	64,167	0	68,326
amount	72,795	0	68,156	0	72,795
tax	76,398	0	71,312	0	76,398
involv	81,131	0	75,498	0	81,131
limit	91,816	0	85,179	0	91,816
major	95,810	0	88,610	0	95,810
world	96,436	0	88,913	0	96,436
rate	118,907	0	109,292	0	118,907
govern	202,082	0	185,164	0	202,082
unit	236,217	0	215,767	0	236,217
countri	237,522	0	216,280	0	237,522
includ	243,806	0	221,306	0	243,806
	2,270,789			**623,867**	**1,646,922**



Percentage Reduction: 72.53%

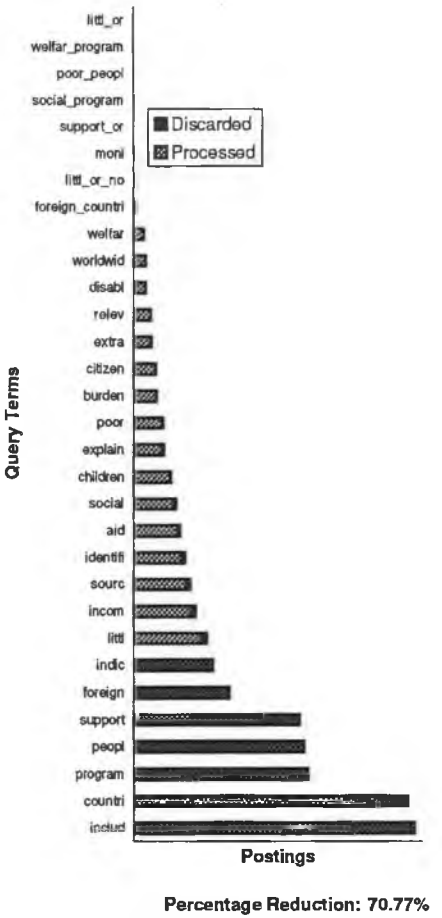
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	407	407
Rel_ret:	**116**	**56**
P. at 0.0	0.1567	0.0976
P. at 0.1	0.156	0.0804
P. at 0.2	0.129	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.038**	**0.0101**
P @ 10 D.	0	0
P @ 30 D.	0.0667	0.0667
P @ 100 D.	0.12	0.08



	No QSR	QSR	% Red.
Seconds:	34.8	10.1	70.98%
Doc. Acc:	809,266	50,000	93.82%

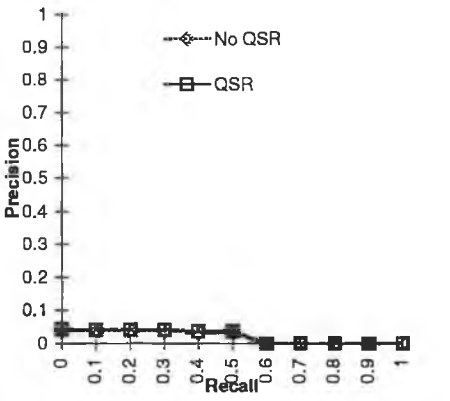
Query: 292

TERM	NP	QTT	PLT	Processed	Discarded
littl_or	254	1	254	254	0
welfar_program	278	1	278	278	0
poor_peopl	551	1	551	551	0
social_program	584	1	584	584	0
support_or	748	1	745	745	3
moni	1,219	1	1,211	1,211	8
littl_or_no	1,676	1	1,659	1,659	17
foreign_countri	2,285	1	2,254	2,254	31
welfar	9,080	1	8,926	8,926	154
worldwid	10,488	1	10,274	10,274	214
disabl	10,534	1	10,283	10,283	251
relev	15,178	1	14,764	14,764	414
extra	15,725	1	15,242	15,242	483
citizen	19,518	1	18,851	18,851	667
burden	20,018	1	19,265	19,265	753
poor	25,388	1	24,346	24,346	1,042
explain	26,304	1	25,133	25,133	1,171
children	32,397	1	30,844	30,844	1,553
social	36,468	1	34,594	34,594	1,874
aid	40,212	1	38,007	38,007	2,205
identifi	44,281	1	41,700	41,700	2,581
sourc	48,725	1	45,717	45,717	3,008
incom	53,377	1	49,898	49,898	3,479
littl	63,268	1	58,926	58,926	4,342
indic	68,326	0	63,401	0	68,326
foreign	82,835	0	76,579	0	82,835
support	144,127	0	132,745	0	144,127
peopl	147,661	0	135,491	0	147,661
program	151,200	0	138,217	0	151,200
countri	237,522	0	216,308	0	237,522
includ	243,806	0	221,190	0	243,806
	1,554,033			**454,306**	**1,099,727**



Percentage Reduction: 70.77%

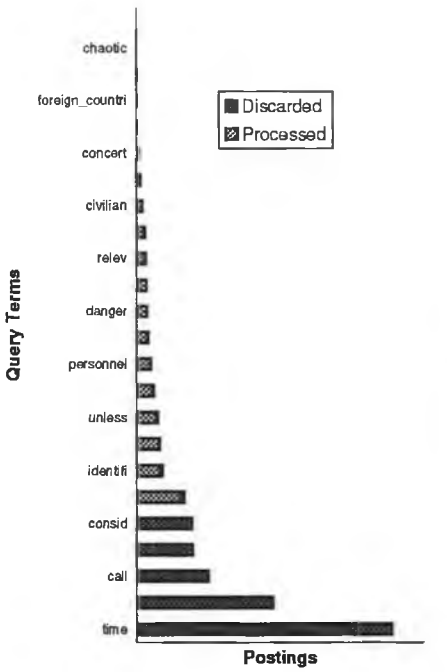
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	59	59
Rel_ret:	**30**	**31**
P. at 0.0	0.0349	0.0423
P. at 0.1	0.0344	0.042
P. at 0.2	0.0344	0.042
P. at 0.3	0.0344	0.042
P. at 0.4	0.0315	0.037
P. at 0.5	0.0305	0.0368
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0147**	**0.0173**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0.03	0.03



	No QSR	QSR	% Red.
Seconds:	24.6	8.3	66.26%
Doc. Acc:	738,092	50,000	93.23%

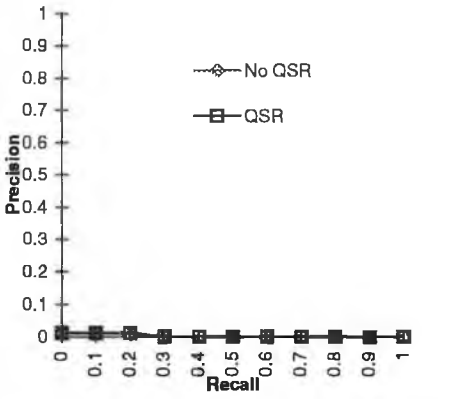
Query: 293

TERM	NP	QTT	PLT	Processed	Discarded
danger_situat	135	1	135	135	0
chaotic	742	1	742	742	0
militari_personnel	1,768	1	1,768	1,768	0
foreign_countri	2,285	1	2,279	2,279	6
evacu	2,718	1	2,699	2,699	19
concert	4,971	1	4,914	4,914	57
combat	6,830	1	6,723	6,723	107
civilian	10,016	1	9,815	9,815	201
accomplish	14,126	1	13,782	13,782	344
relev	15,178	1	14,742	14,742	436
instanc	17,111	1	16,545	16,545	566
danger	18,354	1	17,667	17,667	687
citizen	19,518	1	18,703	18,703	815
personnel	24,729	1	23,589	23,589	1,140
situat	29,825	1	28,320	28,320	1,505
unless	36,502	1	34,502	34,502	2,000
militari	40,513	1	38,117	38,117	2,396
identifi	44,281	1	41,470	41,470	2,811
foreign	82,835	1	77,216	77,216	5,619
consid	95,644	0	88,740	0	95,644
world	96,436	0	89,056	0	96,436
call	124,865	0	114,767	0	124,865
countri	237,522	0	217,280	0	237,522
time	443,964	0	404,200	0	443,964
	1,370,868			**353,728**	**1,017,140**



Percentage Reduction: 74.20%

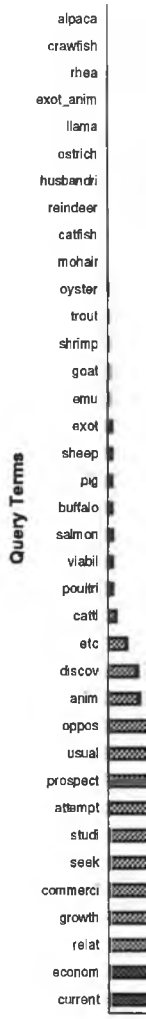
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	41	41
Rel_ret:	**9**	**9**
P. at 0.0	0.0115	0.0119
P. at 0.1	0.0115	0.0119
P. at 0.2	0.0108	0.0119
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.002**	**0.002**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0	0



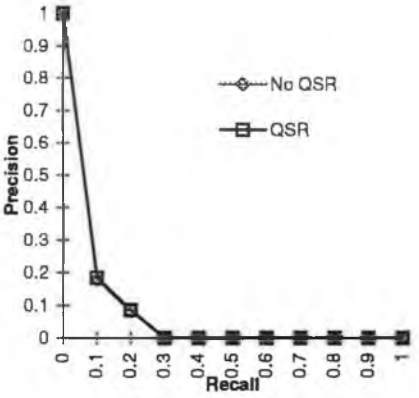
	No QSR	QSR	% Red.
Seconds:	21.8	6.9	68.35%
Doc. Acc:	709,656	50,000	92.95%

Query: 294

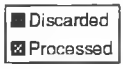
TERM	NP	QTT	PLT	Processed	Discarded
crawfish	39	1	39	39	0
alpaca	39	1	39	39	0
rhea	42	1	42	42	0
exot_anim	65	1	65	65	0
llama	66	1	65	65	1
ostrich	105	1	104	104	1
husbandri	114	1	113	113	1
reindeer	126	1	124	124	2
catfish	128	1	126	126	2
mohair	238	1	234	234	4
oyster	414	1	406	406	8
trout	631	1	617	617	14
shrimp	665	1	649	649	16
goat	877	1	853	853	24
emu	958	1	929	929	29
exot	1,577	1	1,526	1,526	51
sheep	1,617	1	1,560	1,560	57
pig	1,638	1	1,575	1,575	63
buffalo	1,732	1	1,661	1,661	71
salmon	1,852	1	1,771	1,771	81
viabil	1,892	1	1,803	1,803	89
poultri	1,918	1	1,823	1,823	95
cattl	3,010	1	2,852	2,852	158
etc	7,173	1	6,777	6,777	396
discov	11,337	1	10,679	10,679	658
anim	12,100	1	11,363	11,363	737
oppos	22,632	1	21,190	21,190	1,442
usual	24,156	1	22,547	22,547	1,609
prospect	28,482	1	26,504	26,504	1,978
attempt	39,644	1	36,778	36,778	2,866
studi	43,343	1	40,086	40,086	3,257
seek	49,003	1	45,180	45,180	3,823
commerci	51,581	1	47,410	47,410	4,171
growth	54,553	1	49,986	49,986	4,567
relat	95,201	1	86,959	86,959	8,242
econom	106,546	0	97,017	0	106,546
current	112,595	0	102,204	0	112,595
	678,089			**424,435**	**253,654**

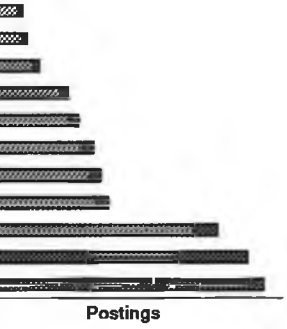


	No QSR	QSR
Retrieved:	1000	1000
Relevant:	160	160
Rel_ret:	**42**	**42**
P. at 0.0	1	1
P. at 0.1	0.1789	0.1848
P. at 0.2	0.0831	0.086
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0529**	**0.0533**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.2333	0.2333
P @ 100 D.	0.17	0.17



	No QSR	QSR	% Red.
Seconds:	15.6	9	42.31%
Doc. Acc:	460,013	50,000	89.13%

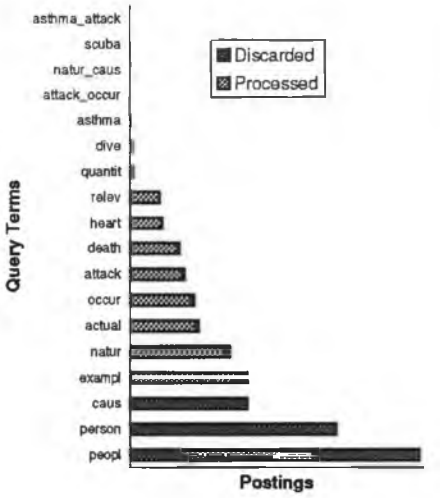




Percentage Reduction: 37.41%

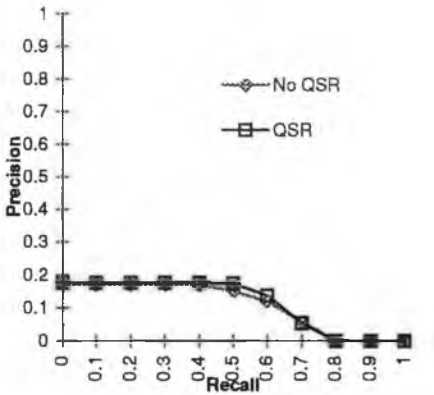
Query: 295

TERM	NP	QTT	PLT	Processed	Discarded
asthma_attack	28	1	28	28	0
scuba	127	1	127	127	0
natur_caus	152	1	151	151	1
attack_occur	159	1	157	157	2
asthma	498	1	491	491	7
dive	1,326	1	1,302	1,302	24
quantit	1,745	1	1,704	1,704	41
relev	15,178	1	14,739	14,739	439
heart	16,487	1	15,919	15,919	568
death	25,116	1	24,111	24,111	1,005
attack	27,898	1	26,627	26,627	1,271
occur	32,742	1	31,068	31,068	1,674
actual	34,904	1	32,926	32,926	1,978
natur	51,136	1	47,954	47,954	3,182
exampl	59,723	1	55,675	55,675	4,048
caus	59,891	0	55,498	0	59,891
person	105,216	0	96,915	0	105,216
peopl	147,661	0	135,191	0	147,661
	579,987			**252,979**	**327,008**



Percentage Reduction: 56.38%

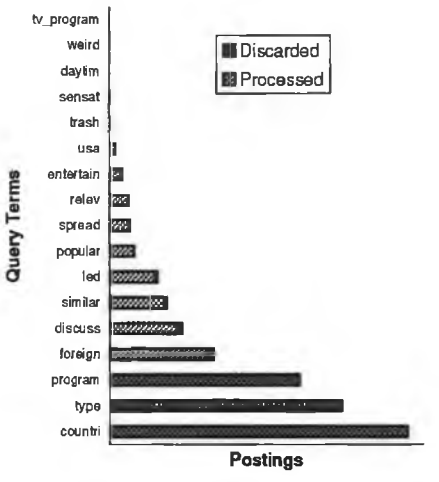
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	15	15
Rel_ret:	**11**	**11**
P. at 0.0	0.1714	0.1765
P. at 0.1	0.1714	0.1765
P. at 0.2	0.1714	0.1765
P. at 0.3	0.1714	0.1765
P. at 0.4	0.1714	0.1765
P. at 0.5	0.1509	0.1739
P. at 0.6	0.12	0.1385
P. at 0.7	0.0591	0.0529
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0866**	**0.0915**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.0667	0.0667
P @ 100 D.	0.09	0.09



	No QSR	QSR	% Red.
Seconds:	13.2	6	54.55%
Doc. Acc:	409,572	50,000	87.79%

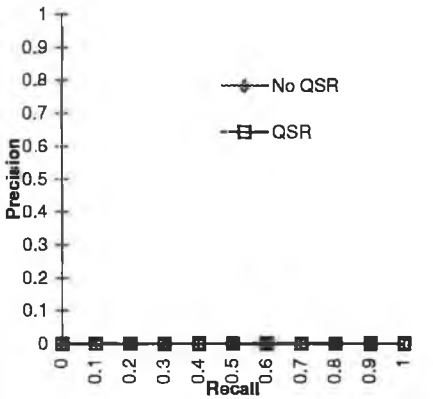
Query: 296

TERM	NP	QTT	PLT	Processed	Discarded
tv_program	203	1	203	203	0
weird	469	1	469	469	0
daytim	527	1	526	526	1
sensat	769	1	763	763	6
trash	1,370	1	1,351	1,351	19
usa	4,587	1	4,497	4,497	90
entertain	9,973	1	9,720	9,720	253
relev	15,178	1	14,704	14,704	474
spread	16,276	1	15,672	15,672	604
popular	19,650	1	18,806	18,806	844
led	37,955	1	36,101	36,101	1,854
similar	45,918	1	43,406	43,406	2,512
discuss	58,190	1	54,664	54,664	3,526
foreign	82,835	1	77,328	77,328	5,507
program	151,200	0	140,260	0	151,200
type	184,776	0	170,319	0	184,776
countri	237,522	0	217,542	0	237,522
	867,398			**278,210**	**589,188**



Percentage Reduction: 67.93%

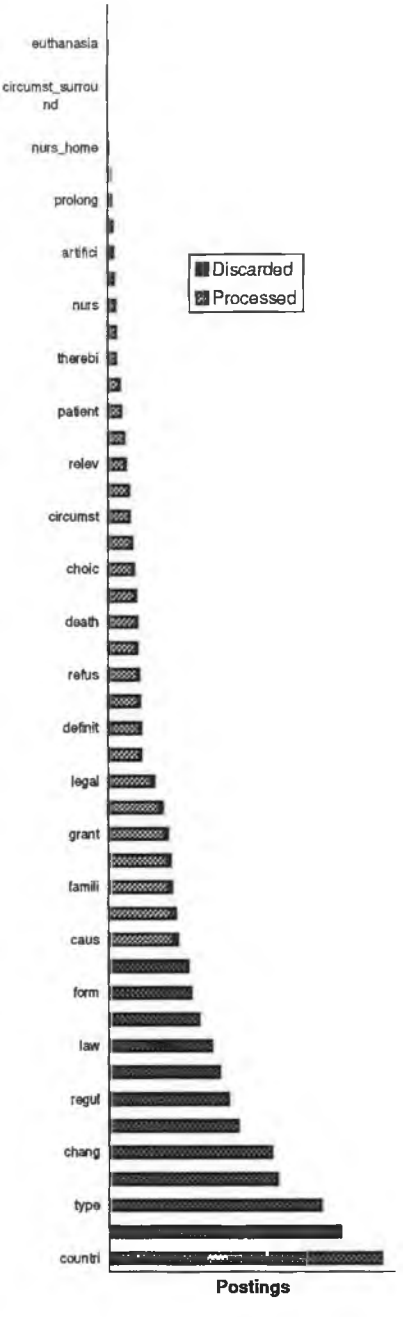
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	1	1
Rel_ret:	**0**	**0**
P. at 0.0	0	0
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0**	**0**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0	0



	No QSR	QSR	% Red.
Seconds:	16.3	6	63.19%
Doc. Acc:	536,148	50,000	90.67%

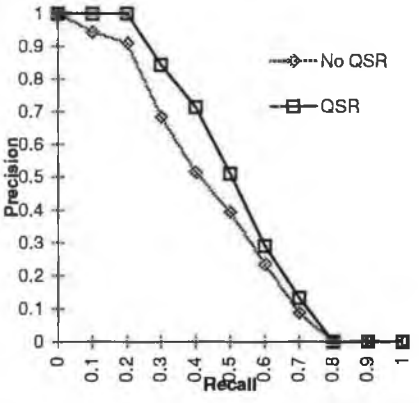
Query: 297

TERM	NP	QTT	PLT	Processed	Discarded
legal_and_ethic	60	1	60	60	0
euthanasia	108	1	108	108	0
caus_death	167	1	167	167	0
circumst_surround	332	1	332	332	0
ordeal	649	1	649	649	0
nurs_home	1,429	1	1,428	1,428	1
regul_govern	2,172	1	2,166	2,166	6
prolong	3,048	1	3,033	3,033	15
discontinu	3,733	1	3,706	3,706	27
artifici	4,364	1	4,323	4,323	41
ethic	4,951	1	4,893	4,893	58
nurs	6,094	1	6,010	6,010	84
die	6,538	1	6,433	6,433	105
therebi	6,789	1	6,665	6,665	124
surround	10,008	1	9,803	9,803	205
patient	11,277	1	11,021	11,021	256
reveal	14,195	1	13,841	13,841	354
relev	15,178	1	14,766	14,766	412
articl	17,829	1	17,306	17,306	523
circumst	18,469	1	17,886	17,886	583
pro	20,664	1	19,966	19,966	698
choic	22,615	1	21,800	21,800	815
treatment	24,195	1	23,270	23,270	925
death	25,116	1	24,100	24,100	1,016
throughout	25,226	1	24,149	24,149	1,077
especi	28,553	1	27,080	27,080	1,473
legal	39,598	1	37,468	37,468	2,130
reflect	46,995	1	44,363	44,363	2,632
grant	51,486	1	48,488	48,488	2,998
life	53,933	1	50,673	50,673	3,260
famili	55,070	1	51,618	51,618	3,452
accept	58,356	1	54,569	54,569	3,787
caus	59,891	1	55,871	55,871	4,020
home	68,639	0	63,880	0	68,639
form	71,419	0	66,308	0	71,419
mean	78,434	0	72,647	0	78,434
law	89,340	0	82,550	0	89,340
world	96,436	0	88,892	0	96,436
regul	104,038	0	95,668	0	104,038
current	112,595	0	103,287	0	112,595
chang	141,578	0	129,559	0	141,578
inform	146,656	0	133,880	0	146,656
type	184,776	0	168,269	0	184,776
govern	202,082	0	183,580	0	202,082
countri	237,522	0	215,247	0	237,522
	2,255,139			**686,654**	**1,568,485**



Percentage Reduction: 69.55%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	86	86
Rel_ret:	**63**	**64**
P. at 0.0	1	1
P. at 0.1	0.9444	1
P. at 0.2	0.9091	1
P. at 0.3	0.6842	0.8438
P. at 0.4	0.5147	0.7143
P. at 0.5	0.3929	0.5116
P. at 0.6	0.2353	0.2905
P. at 0.7	0.0875	0.1328
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.4166**	**0.4894**
P @ 10 D.	0.9	1
P @ 30 D.	0.7667	0.8333
P @ 100 D.	0.41	0.44



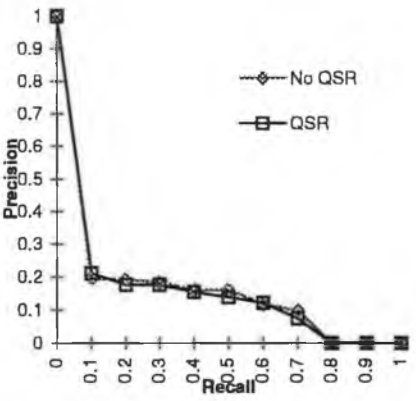
	No QSR	QSR	% Red.
Seconds:	35	11.8	66.29%
Doc. Acc:	814,654	50,000	93.86%

Query: 298

TERM	NP	QTT	PLT	Processed	Discarded
control_measur	1,118	1	1,118	1,118	0
control_i	1,374	1	1,374	1,374	0
violent_crime	1,379	1	1,379	1,379	0
strict	4,052	1	4,030	4,030	22
violent	6,088	1	6,023	6,023	65
gun	7,919	1	7,793	7,793	126
controversi	12,818	1	12,548	12,548	270
crime	15,512	1	15,103	15,103	409
answer	22,221	1	21,519	21,519	702
enforc	28,673	1	27,616	27,616	1,057
inspect	29,184	1	27,955	27,955	1,229
restrict	31,909	1	30,397	30,397	1,512
experi	32,994	1	31,257	31,257	1,737
measur	56,018	1	52,774	52,774	3,244
question	65,036	1	60,928	60,928	4,108
reduc	79,858	0	74,394	0	79,858
period	92,277	0	85,477	0	92,277
control	96,879	0	89,230	0	96,879
effect	135,381	0	123,980	0	135,381
countri	237,522	0	216,270	0	237,522
	958,212			**301,814**	**656,398**

Percentage Reduction: 68.50%

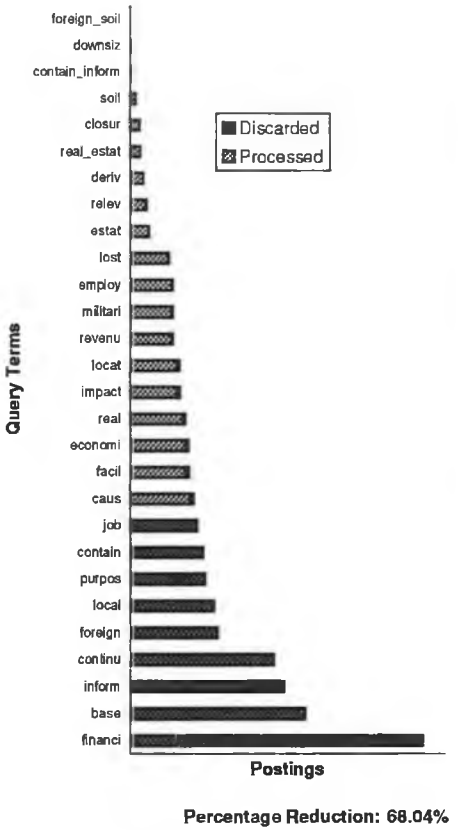
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	91	91
Rel_ret:	**70**	**67**
P. at 0.0	1	1
P. at 0.1	0.197	0.2128
P. at 0.2	0.1935	0.1765
P. at 0.3	0.184	0.1765
P. at 0.4	0.161	0.1561
P. at 0.5	0.161	0.1391
P. at 0.6	0.1184	0.122
P. at 0.7	0.0988	0.0739
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1554**	**0.1407**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.2667	0.2667
P @ 100 D.	0.18	0.13



	No QSR	QSR	% Red.
Seconds:	17.8	6.3	64.61%
Doc. Acc:	599,743	50,000	91.66%

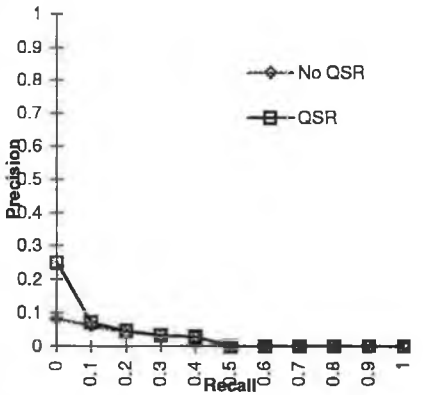
Query: 299

TERM	NP	QTT	PLT	Processed	Discarded
foreign_soil	46	1	46	46	0
downsiz	1,157	1	1,157	1,157	0
contain_inform	1,453	1	1,453	1,453	0
soil	4,928	1	4,924	4,924	4
closur	8,642	1	8,603	8,603	39
real_estat	9,402	1	9,325	9,325	77
deriv	12,165	1	12,020	12,020	145
relev	15,178	1	14,941	14,941	237
estat	17,583	1	17,244	17,244	339
lost	36,639	1	35,797	35,797	842
employ	40,336	1	39,260	39,260	1,076
militari	40,513	1	39,282	39,282	1,231
revenu	40,527	1	39,146	39,146	1,381
locat	46,533	1	44,775	44,775	1,758
impact	46,830	1	44,887	44,887	1,943
real	51,916	1	49,570	49,570	2,346
economi	55,130	1	52,434	52,434	2,696
facil	55,698	1	52,768	52,768	2,930
caus	59,891	1	56,519	56,519	3,372
job	63,608	0	59,791	0	63,608
contain	69,251	0	64,839	0	69,251
purpos	70,718	0	65,951	0	70,718
local	79,412	0	73,764	0	79,412
foreign	82,835	0	76,637	0	82,835
continu	136,778	0	126,038	0	136,778
inform	146,656	0	134,597	0	146,656
base	166,964	0	152,617	0	166,964
financi	279,312	0	254,277	0	279,312
	1,640,101			**524,151**	**1,115,950**



Percentage Reduction: 68.04%

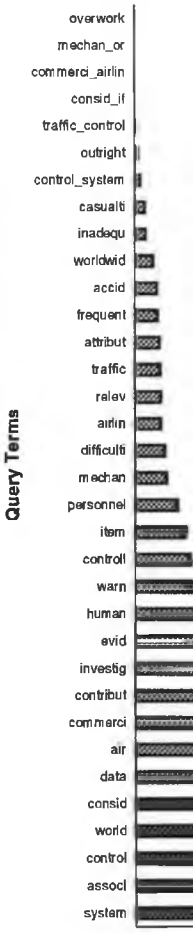
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	62	62
Rel_ret:	**27**	**27**
P. at 0.0	0.0833	0.25
P. at 0.1	0.0613	0.0714
P. at 0.2	0.0431	0.0471
P. at 0.3	0.0332	0.0327
P. at 0.4	0.0283	0.0285
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0181**	**0.0213**
P @ 10 D.	0	0.1
P @ 30 D.	0.0333	0.0333
P @ 100 D.	0.04	0.07



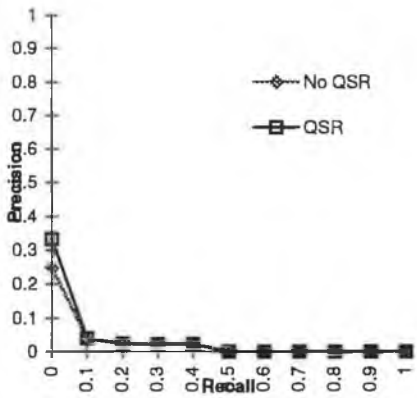
	No QSR	QSR	% Red.
Seconds:	26.2	8.8	66.41%
Doc. Acc:	785,072	50,000	93.63%

Query: 300

TERM	NP	QTT	PLT	Processed	Discarded
overwork	236	1	236	236	0
mechan_or	278	1	278	278	0
commerci_airlin	337	1	337	337	0
consid_if	400	1	400	400	0
traffic_control	1,178	1	1,175	1,175	3
outright	2,353	1	2,341	2,341	12
control_system	3,330	1	3,302	3,302	28
casualti	5,894	1	5,827	5,827	67
inadequ	6,188	1	6,099	6,099	89
worldwid	10,488	1	10,304	10,304	184
accid	12,941	1	12,674	12,674	267
frequent	13,299	1	12,983	12,983	316
attribut	14,214	1	13,831	13,831	383
traffic	14,700	1	14,259	14,259	441
relev	15,178	1	14,675	14,675	503
airlin	15,206	1	14,654	14,654	552
difficulti	17,263	1	16,583	16,583	680
mechan	18,532	1	17,744	17,744	788
personnel	24,729	1	23,600	23,600	1,129
item	29,027	1	27,611	27,611	1,416
controll	31,957	1	30,299	30,299	1,658
warn	32,382	1	30,600	30,600	1,782
human	36,323	1	34,211	34,211	2,112
evid	37,806	1	35,490	35,490	2,316
investig	38,704	1	36,212	36,212	2,492
contribut	39,185	1	36,540	36,540	2,645
commerci	51,581	1	47,938	47,938	3,643
air	53,654	1	49,697	49,697	3,957
data	84,903	1	78,376	78,376	6,527
consid	95,644	0	87,992	0	95,644
world	96,436	0	88,419	0	96,436
control	96,879	0	88,523	0	96,879
associ	115,644	0	105,308	0	115,644
system	167,317	0	151,840	0	167,317
	1,184,186			**578,276**	**605,910**



	No QSR	QSR
Retrieved:	1000	1000
Relevant:	44	44
Rel_ret:	19	20
P. at 0.0	0.25	0.3333
P. at 0.1	0.0391	0.0403
P. at 0.2	0.022	0.0244
P. at 0.3	0.0207	0.0226
P. at 0.4	0.0205	0.0226
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0179**	**0.0224**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.0667	0.1
P @ 100 D	0.03	0.03



	No QSR	QSR	% Red.
Seconds:	19.3	8.5	55.96%
Doc. Acc:	637,537	50,000	92.16%

Percentage Reduction: 51.17%

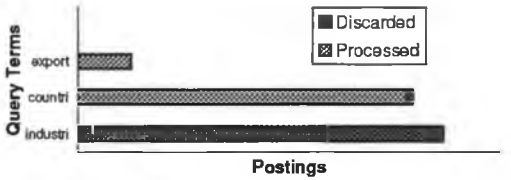
Appendix B

This Appendix contains results on a per query basis for the *TREC-5* automatic submission using the optimal settings obtained from the *TREC-4* experimental runs. The results presented for each query (251 to 300) are as follows:

- A tabular description of the Query Space.
- A graphical description of the Query Space.
- A comparison of effectiveness between the query with QSR switched on and the query with QSR switched off.
- A comparison of efficiency between the query with QSR switched on and the query with QSR switched off.

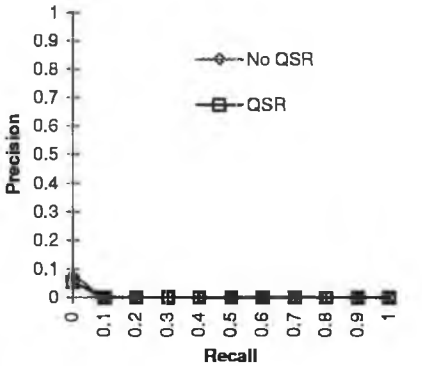
Query: 251

Query Term	NP	QTT	PLT	Processed	Discarded
export	37741	1	37741	37,741	0
countri	237522	1	233365	233,365	4,157
industri	259266	0	248247	0	259,266
	534,529			**271,106**	**263,423**



Percentage Reduction: 49.28%

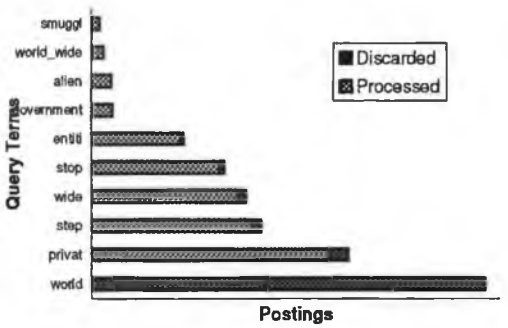
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	579	579
Rel_ret:	**52**	**52**
P. at 0.0	0.0769	0.0542
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0051**	**0.0042**
P @ 10 D.	0	0
P @ 30 D.	0.067	0
P @ 100 D.	0.06	0.02



	No QSR	QSR	% Red.
Seconds:	12.4	5.8	53.23%
Doc. Acc:	352,015	50,000	85.80%

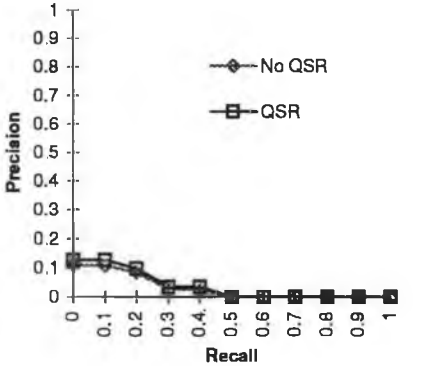
Query: 252

TERM	NP	QTT	PLT	Processed	Discarded
smuggl	2051	1	2051	2,051	0
world_wide	2935	1	2935	2,935	0
alien	4895	1	4846	4,846	49
government	5237	1	5132	5,132	105
entiti	22570	1	21892	21,892	678
stop	32448	1	31150	31,150	1,298
wide	37694	1	35809	35,809	1,885
step	41546	1	39053	39,053	2,493
privat	62828	1	58430	58,430	4,398
world	96436	0	88721	0	96,436
	308,640			**201,298**	**107,342**



Percentage Reduction: 34.78%

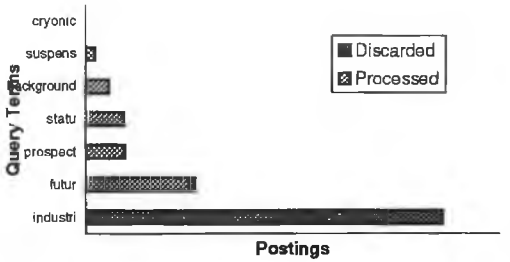
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	37	37
Rel_ret:	**18**	**18**
P. at 0.0	0.1094	0.1273
P. at 0.1	0.1094	0.1273
P. at 0.2	0.0826	0.0964
P. at 0.3	0.0276	0.0342
P. at 0.4	0.0258	0.0339
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0286**	**0.0328**
P @ 10 D.	0	0
P @ 30 D.	0.067	0.067
P @ 100 D.	0.07	0.09



	No QSR	QSR	% Red.
Seconds:	10.1	4.5	55.45%
Doc. Acc:	255,637	50,000	80.44%

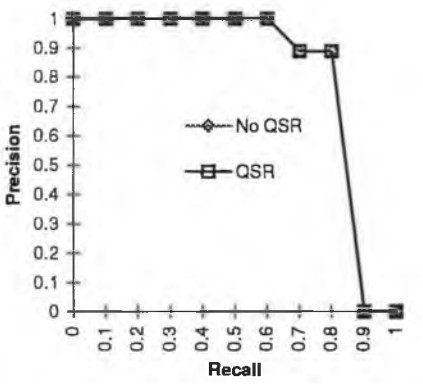
Query: 253

TERM	NP	QTT	PLT	Processed	Discarded
cryonic	8	1	8	8	0
suspens	6892	1	6866	6,866	26
background	16815	1	16541	16,541	274
statu	28029	1	27223	27,223	806
prospect	28482	1	27307	27,307	1,175
futur	79462	1	75190	75,190	4,272
industri	259266	0	242089	0	259,266
	418,954			**153,135**	**265,819**



Percentage Reduction: 63.45%

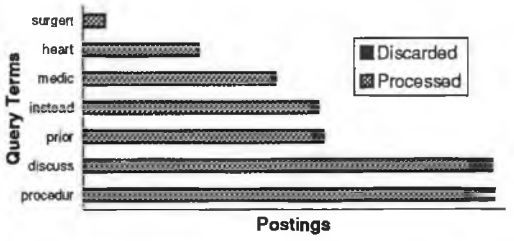
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	10	10
Rel_ret:	**8**	**8**
P. at 0.0	1	1
P. at 0.1	1	1
P. at 0.2	1	1
P. at 0.3	1	1
P. at 0.4	1	1
P. at 0.5	1	1
P. at 0.6	1	1
P. at 0.7	0.8889	0.8889
P. at 0.8	0.8889	0.8889
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.7764**	**0.7764**
P @ 10 D.	0.8	0.8
P @ 30 D.	0.267	0.267
P @ 100 D.	0.08	0.08



	No QSR	QSR	% Red.
Seconds:	12.2	4	67.21%
Doc. Acc:	363,934	50,000	86.26%

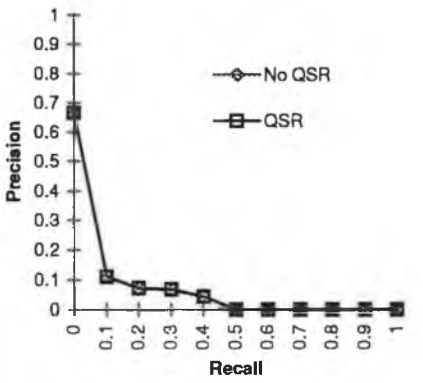
Query: 254

TERM	NP	QTT	PLT	Processed	Discarded
surgeri	3121	1	3121	3,121	0
heart	16487	1	16425	16,425	62
medic	27411	1	26965	26,965	446
instead	33519	1	32555	32,555	964
prior	34222	1	32810	32,810	1,412
discuss	58190	1	55062	55,062	3,128
procedur	58527	1	54649	54,649	3,878
	231,477			**221,587**	**9,890**



Percentage Reduction: 4.27%

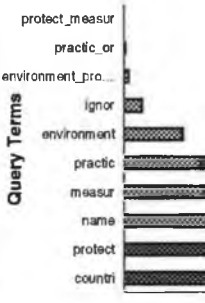
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	85	85
Rel_ret:	**39**	**39**
P. at 0.0	0.6667	0.6667
P. at 0.1	0.1098	0.1098
P. at 0.2	0.0717	0.0723
P. at 0.3	0.0662	0.0674
P. at 0.4	0.0435	0.0442
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0527**	**0.0531**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.2	0.2
P @ 100 D.	0.09	0.09



	No QSR	QSR	% Red.
Seconds:	8.3	5.4	34.94%
Doc. Acc:	199,519	50,000	74.94%

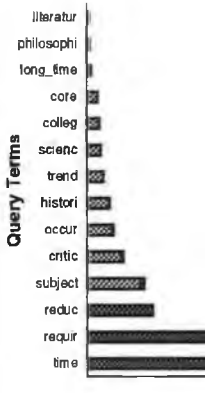
Query: 255

TERM	NP	QTT	PLT	Processed	Discarded
protect_measur	408	1	408	408	0
practic_or	713	1	713	713	0
nvironment_protect	3008	1	2977	2,977	31
ignor	12007	1	11766	11,766	241
environment	39337	1	38156	38,156	1,181
practic	52853	1	50738	50,738	2,115
measur	56018	1	53217	53,217	2,801
name	66610	1	62613	62,613	3,997
protect	72331	0	67267	0	72,331
countri	237522	0	218520	0	237,522
	540,807			**220,588**	**320,219**



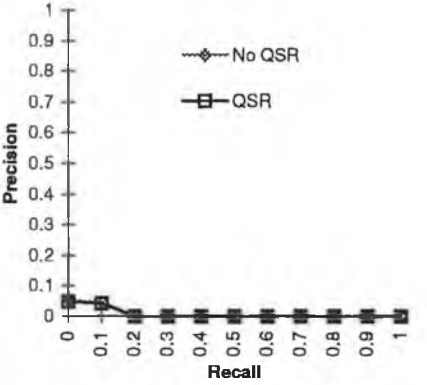
Query: 256

TERM	NP	QTT	PLT	Processed	Discarded
literatur	2974	1	2974	2,974	0
philosophi	3713	1	3713	3,713	0
long_time	5081	1	5059	5,059	22
core	13148	1	12997	12,997	151
colleg	15791	1	15497	15,497	294
scienc	17082	1	16642	16,642	440
trend	20435	1	19763	19,763	672
histori	28121	1	26996	26,996	1,125
occur	32742	1	31198	31,198	1,544
critic	44092	1	41698	41,698	2,394
subject	69574	1	65300	65,300	4,274
reduc	79858	0	74382	0	79,858
requir	194580	0	179847	0	194,580
time	443964	0	407178	0	443,964
	971,155			**241,837**	**729,318**



Percentage Reduction: 59.21%

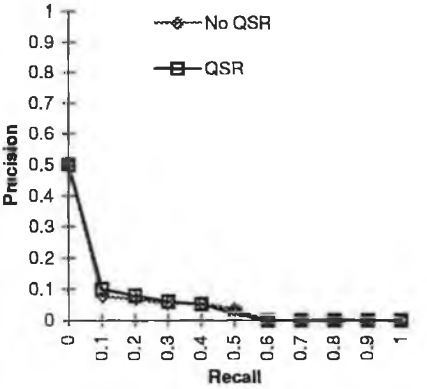
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	109	109
Rel_ret:	**21**	**20**
P. at 0.0	0.0526	0.0476
P. at 0.1	0.0453	0.0427
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0075**	**0.0067**
P @ 10 D.	0	0
P @ 30 D.	0	0.033
P @ 100 D	0.03	0.02



	No QSR	QSR	% Red.
Seconds:	10.9	5	54.13%
Doc. Acc:	432,511	50,000	88.44%

Percentage Reduction: 75.10%

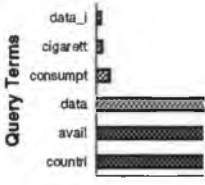
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	22	22
Rel_ret:	**13**	**13**
P. at 0.0	0.5	0.5
P. at 0.1	0.0769	0.1
P. at 0.2	0.0698	0.0794
P. at 0.3	0.0548	0.0603
P. at 0.4	0.0536	0.052
P. at 0.5	0.0354	0.022
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0518**	**0.0544**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.067	0.1
P @ 100 D.	0.06	0.06



	No QSR	QSR	% Red.
Seconds:	16.9	5.5	67.46%
Doc. Acc:	500,000	50,000	90.00%

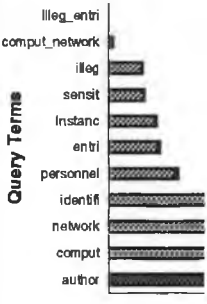
Query: 257

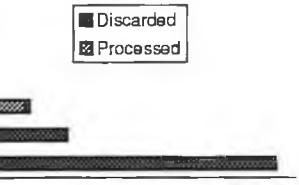
TERM	NP	QTT	PLT	Processed	Discarded
data_i	2595	1	2595	2,595	0
cigarett	3465	1	3445	3,445	20
consumpt	8170	1	8006	8,006	164
data	84903	1	81992	81,992	2,911
avail	108167	0	102913	0	108,167
countri	237522	0	222592	0	237,522
	444,822			**96,038**	**348,784**



Query: 258

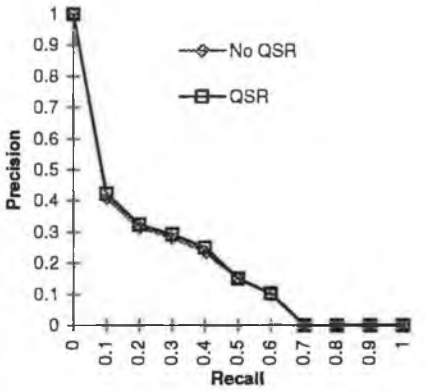
TERM	NP	QTT	PLT	Processed	Discarded
illeg_entri	62	1	62	62	0
comput_network	1480	1	1480	1,480	0
illeg	11958	1	11860	11,860	98
sensit	12815	1	12593	12,593	222
instanc	17111	1	16659	16,659	452
entri	18166	1	17521	17,521	645
personnel	24729	1	23627	23,627	1,102
identifi	44281	1	41905	41,905	2,376
network	44580	1	41783	41,783	2,797
comput	74133	1	68808	68,808	5,325
author	125196	0	115066	0	125,196
	374,511			**236,298**	**138,213**



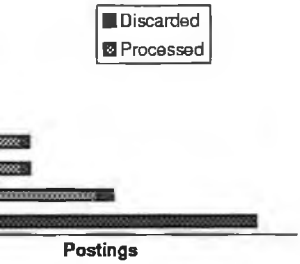


Percentage Reduction: 78.41%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	135	135
Rel_ret:	83	83
P. at 0.0	1	1
P. at 0.1	0.4103	0.4242
P. at 0.2	0.314	0.3241
P. at 0.3	0.2838	0.2941
P. at 0.4	0.2358	0.25
P. at 0.5	0.1509	0.1498
P. at 0.6	0.1021	0.1024
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.1923	0.1993
P @ 10 D.	0.6	0.8
P @ 30 D.	0.433	0.433
P @ 100 D.	0.29	0.31

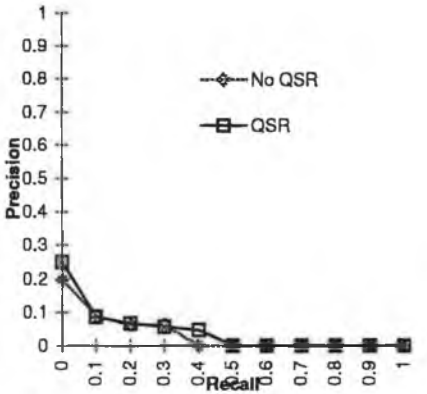


	No QSR	QSR	% Red.
Seconds:	9.8		100.00%
Doc. Acc:	390,482	50,000	87.20%



Percentage Reduction: 36.90%

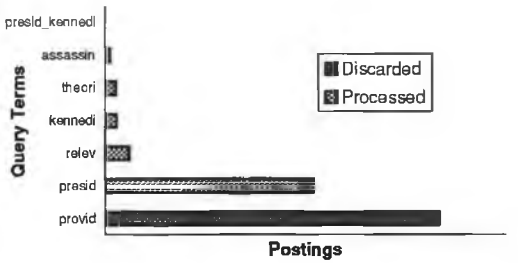
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	115	115
Rel_ret:	45	46
P. at 0.0	0.2	0.25
P. at 0.1	0.087	0.085
P. at 0.2	0.063	0.0658
P. at 0.3	0.0619	0.0566
P. at 0.4	0	0.0471
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.031	0.0337
P @ 10 D.	0.2	0.2
P @ 30 D.	0.167	0.2
P @ 100 D.	0.08	0.08



	No QSR	QSR	% Red.
Seconds:	9.4	5.3	43.62%
Doc. Acc:	303,625	50,000	83.53%

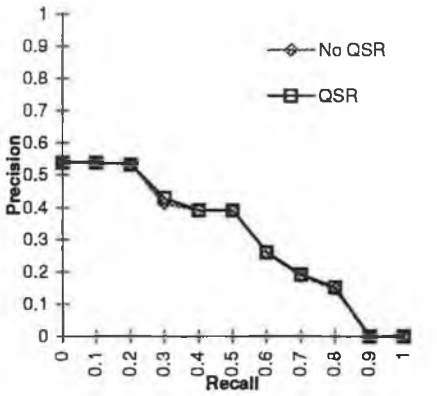
Query: 259

TERM	NP	QTT	PLT	Processed	Discarded
presid_kennedi	399	1	399	399	0
assassin	2997	1	2985	2,985	12
theori	6590	1	6482	6,482	108
kennedi	7333	1	7122	7,122	211
relev	15178	1	14551	14,551	627
presid	130185	1	123187	123,187	6,998
provid	208689	0	194863	0	208,689
	371,371			**154,726**	**216,645**



Percentage Reduction: 58.34%

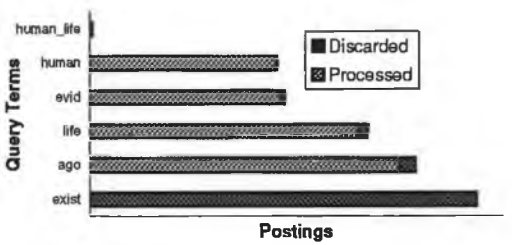
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	36	36
Rel_ret:	**31**	**31**
P. at 0.0	0.5385	0.5385
P. at 0.1	0.5385	0.5385
P. at 0.2	0.5333	0.5333
P. at 0.3	0.4138	0.4286
P. at 0.4	0.3913	0.3913
P. at 0.5	0.3913	0.3913
P. at 0.6	0.2588	0.2588
P. at 0.7	0.1871	0.1926
P. at 0.8	0.1487	0.1518
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.294**	**0.2954**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.4	0.4
P @ 100 D	0.22	0.22



	No QSR	QSR	% Red.
Seconds:	9.2	4.8	47.83%
Doc. Acc:	333,427	50,000	85.00%

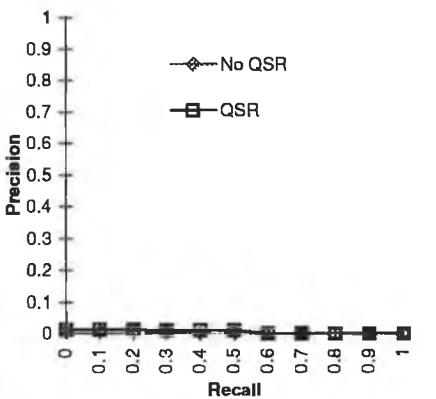
Query: 260

TERM	NP	QTT	PLT	Processed	Discarded
human_life	553	1	553	553	0
human	36323	1	36115	36,115	208
evid	37806	1	37049	37,049	757
life	53933	1	52083	52,083	1,850
ago	63064	1	60000	60,000	3,064
exist	74900	0	70192	0	74,900
	266,579			**185,800**	**80,779**



Percentage Reduction: 30.30%

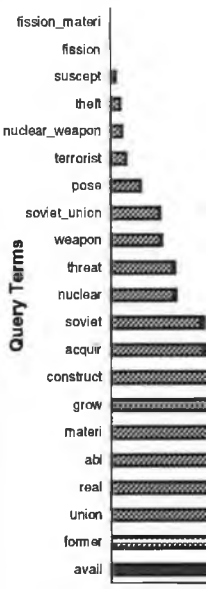
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	22	22
Rel_ret:	**8**	**11**
P. at 0.0	0.0103	0.0114
P. at 0.1	0.0103	0.0114
P. at 0.2	0.0103	0.0114
P. at 0.3	0.0091	0.0113
P. at 0.4	0	0.0113
P. at 0.5	0	0.0113
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0026**	**0.0044**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0	0



	No QSR	QSR	% Red.
Seconds:	8.9	4.5	49.44%
Doc. Acc:	231,237	50,000	78.38%

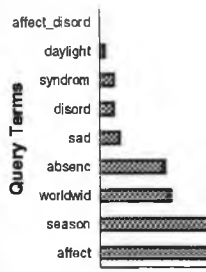
Query: 261

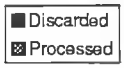
TERM	NP	QTT	PLT	Processed	Discarded
fission_materi	34	1	34	34	0
fission	242	1	242	242	0
suscept	1532	1	1532	1,532	0
theft	2886	1	2873	2,873	13
nuclear_weapon	3602	1	3567	3,567	35
terrorist	4719	1	4650	4,650	69
pose	9180	1	9000	9,000	180
soviet_union	15198	1	14825	14,825	373
weapon	15768	1	15302	15,302	466
threat	19707	1	19027	19,027	680
nuclear	20049	1	19257	19,257	792
soviet	28551	1	27280	27,280	1,271
acquir	29625	1	28158	28,158	1,467
construct	43435	1	41067	41,067	2,368
grow	44028	1	41408	41,408	2,620
materi	44766	1	41878	41,878	2,888
abl	48296	1	44939	44,939	3,357
real	51916	1	48048	48,048	3,868
union	56474	1	51984	51,984	4,490
former	63768	1	58379	58,379	5,389
avail	108167	0	98486	0	108,167
	611,943			**473,450**	**138,493**

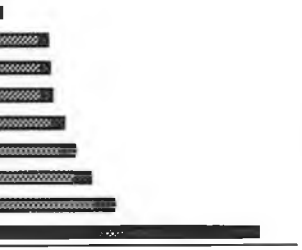


Query: 262

TERM	NP	QTT	PLT	Processed	Discarded
affect_disord	9	1	9	9	0
daylight	736	1	735	735	1
syndrom	2054	1	2031	2,031	23
disord	2106	1	2061	2,061	45
sad	2892	1	2802	2,802	90
absenc	9652	1	9256	9,256	396
worldwid	10488	1	9953	9,953	535
season	18954	1	17797	17,797	1,157
affect	55078	1	51167	51,167	3,911
	101,969			**95,811**	**6,158**



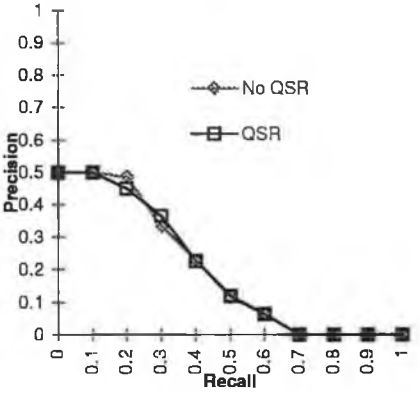




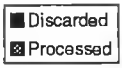
Postings

Percentage Reduction: 22.63%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	87	87
Rel_ret:	**55**	**55**
P. at 0.0	0.5	0.5
P. at 0.1	0.5	0.5
P. at 0.2	0.4865	0.45
P. at 0.3	0.3333	0.3649
P. at 0.4	0.2264	0.2258
P. at 0.5	0.1155	0.1183
P. at 0.6	0.0632	0.0643
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1893**	**0.1905**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.467	0.467
P @ 100 D	0.31	0.32



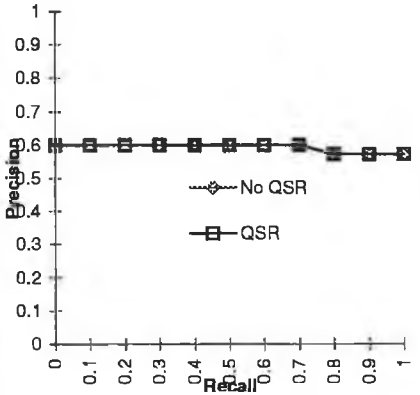
	No QSR	QSR	% Red.
Seconds:	15.3	8.6	43.79%
Doc. Acc:	419,072	50,000	88.07%



Postings

Percentage Reduction: 6.04%

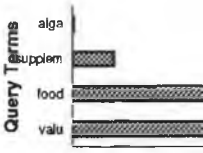
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	4	4
Rel_ret:	**4**	**4**
P. at 0.0	0.6	0.6
P. at 0.1	0.6	0.6
P. at 0.2	0.6	0.6
P. at 0.3	0.6	0.6
P. at 0.4	0.6	0.6
P. at 0.5	0.6	0.6
P. at 0.6	0.6	0.6
P. at 0.7	0.6	0.6
P. at 0.8	0.5714	0.5714
P. at 0.9	0.5714	0.5714
P. at 1.0	0.5714	0.5714
Av. P	**0.5429**	**0.5429**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.133	0.133
P @ 100 D	0.04	0.04



	No QSR	QSR	% Red.
Seconds:	5.7	3.9	31.58%
Doc. Acc:	97,498	50,000	48.72%

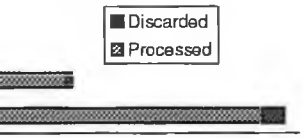
Query: 263

TERM	NP	QTT	PLT	Processed	Discarded
alga	181	1	181	181	0
supplem	7728	1	7635	7,635	93
food	37715	1	36508	36,508	1,207
valu	77228	1	73212	73,212	4,016
	122,852			**117,536**	**5,316**



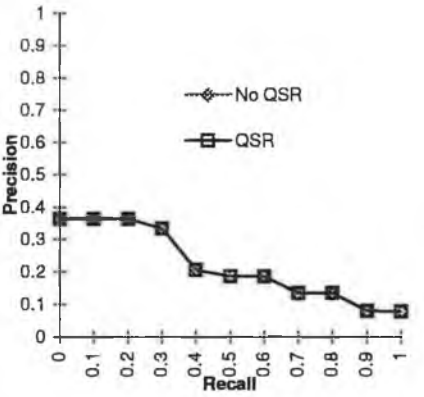
Query: 264

TERM	NP	QTT	PLT	Processed	Discarded
jail	6608	1	6608	6,608	0
instanc	17111	1	17013	17,013	98
citizen	19518	1	19127	19,127	391
identifi	44281	1	42762	42,762	1,519
held	55850	1	53137	53,137	2,713
foreign	82835	1	77628	77,628	5,207
	226,203			**216,275**	**9,928**

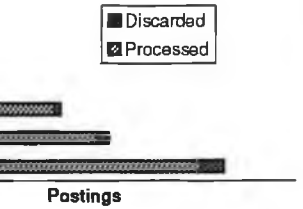


Percentage Reduction: 4.33%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	15	15
Rel_ret:	**15**	**15**
P. at 0.0	0.3636	0.3636
P. at 0.1	0.3636	0.3636
P. at 0.2	0.3636	0.3636
P. at 0.3	0.3333	0.3333
P. at 0.4	0.2069	0.2069
P. at 0.5	0.1875	0.1875
P. at 0.6	0.1875	0.1875
P. at 0.7	0.1327	0.1354
P. at 0.8	0.1327	0.1354
P. at 0.9	0.08	0.0805
P. at 1.0	0.0781	0.0785
Av. P	**0.1968**	**0.1974**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.2	0.2
P @ 100 D.	0.13	0.13

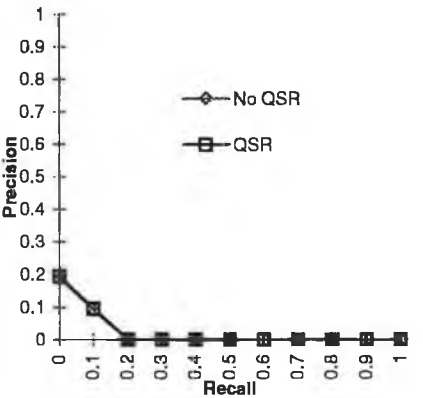


	No QSR	QSR	% Red.
Seconds:	5.5	3.7	32.73%
Doc. Acc:	117,661	50,000	57.51%



Percentage Reduction: 4.39%

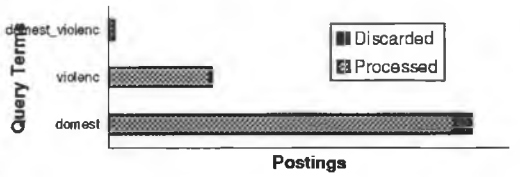
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	281	281
Rel_ret:	**51**	**51**
P. at 0.0	0.1875	0.1935
P. at 0.1	0.0942	0.0954
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0167**	**0.0171**
P @ 10 D.	0	0
P @ 30 D.	0.167	0.167
P @ 100 D.	0.11	0.11



	No QSR	QSR	% Red.
Seconds:	7.9	5	36.71%
Doc. Acc:	202,846	50,000	75.35%

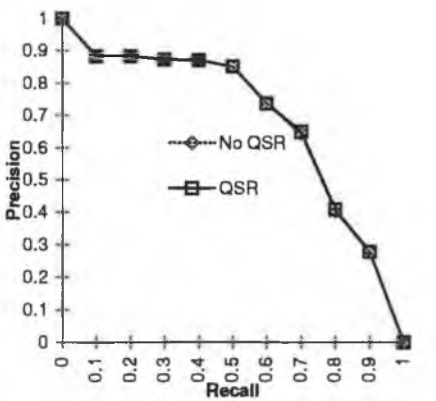
Query: 265

TERM	NP	QTT	PLT	Processed	Discarded
domest_violenc	627	1	627	627	0
violenc	11577	1	11374	11,374	203
domest	40660	1	38931	38,931	1,729
	52,864			**50,932**	**1,932**



Percentage Reduction: 3.65%

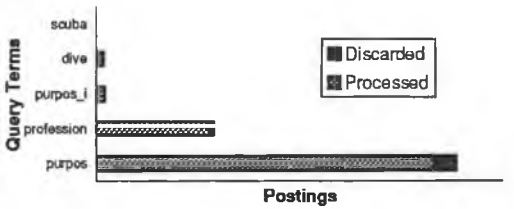
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	147	147
Rel_ret:	**136**	**136**
P. at 0.0	1	1
P. at 0.1	0.8837	0.8837
P. at 0.2	0.8837	0.8837
P. at 0.3	0.8727	0.8727
P. at 0.4	0.8718	0.8718
P. at 0.5	0.8506	0.8506
P. at 0.6	0.7364	0.7364
P. at 0.7	0.6429	0.6485
P. at 0.8	0.4048	0.4089
P. at 0.9	0.2786	0.2771
P. at 1.0	0	0
Av. P	**0.6598**	**0.6595**
P @ 10 D.	0.7	0.7
P @ 30 D.	0.833	0.833
P @ 100 D.	0.81	0.81



	No QSR	QSR	% Red.
Seconds:	3.1	3.1	0.00%
Doc. Acc:	51,269	49,358	3.73%

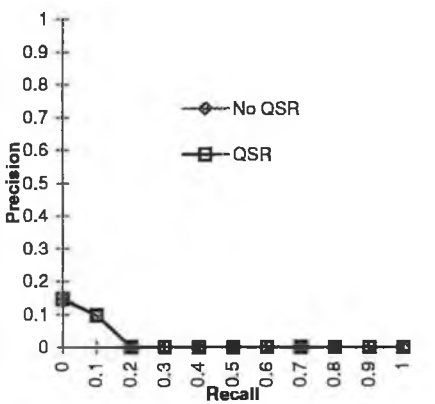
Query: 266

TERM	NP	QTT	PLT	Processed	Discarded
scuba	127	1	127	127	0
dive	1326	1	1314	1,314	12
purpos_i	1627	1	1586	1,586	41
profession	23037	1	22077	22,077	960
purpos	70718	1	66592	66,592	4,126
	96,835			**91,696**	**5,139**



Percentage Reduction: 5.31%

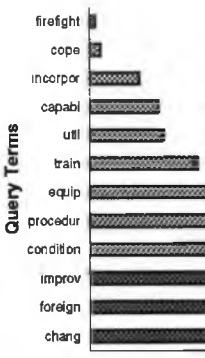
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	139	139
Rel_ret:	**25**	**25**
P. at 0.0	0.1429	0.1471
P. at 0.1	0.0973	0.098
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0161**	**0.0163**
P @ 10 D.	0	0
P @ 30 D.	0.1	0.1
P @ 100 D.	0.09	0.09



	No QSR	QSR	% Red.
Seconds:	4.6	3.6	21.74%
Doc. Acc:	92,701	50,000	46.06%

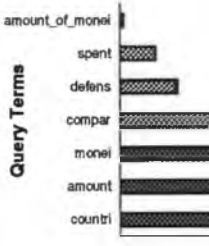
Query: 267

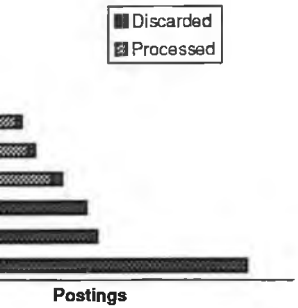
TERM	NP	QTT	PLT	Processed	Discarded
firefight	2262	1	2262	2,262	0
cope	4312	1	4312	4,312	0
incorpor	19647	1	19516	19,516	131
capabl	27179	1	26771	26,771	408
util	29077	1	28398	28,398	679
train	42012	1	40681	40,681	1,331
equip	53183	1	51055	51,055	2,128
procedur	58527	1	55698	55,698	2,829
condition	68845	1	64943	64,943	3,902
improv	78565	0	73458	0	78,565
foreign	82835	0	76760	0	82,835
chang	141578	0	130015	0	141,578
	608,022			**293,636**	**314,386**



Query: 268

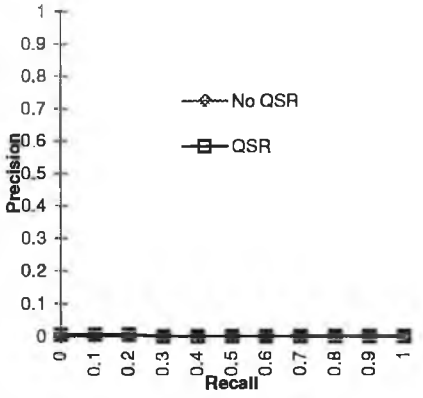
TERM	NP	QTT	PLT	Processed	Discarded
amount_of_monei	2083	1	2083	2,083	0
spent	22777	1	22691	22,691	86
defens	36839	1	36240	36,240	599
compar	60366	1	58630	58,630	1,736
monei	71455	0	68507	0	71,455
amount	72795	0	68882	0	72,795
countri	237522	0	221786	0	237,522
	503,837			**119,644**	**384,193**



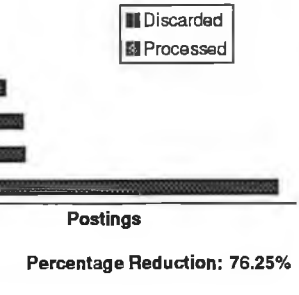


Percentage Reduction: 51.71%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	4	4
Rel_ret:	1	1
P. at 0.0	0.0045	0.0054
P. at 0.1	0.0045	0.0054
P. at 0.2	0.0045	0.0054
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.0011	0.0013
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0	0

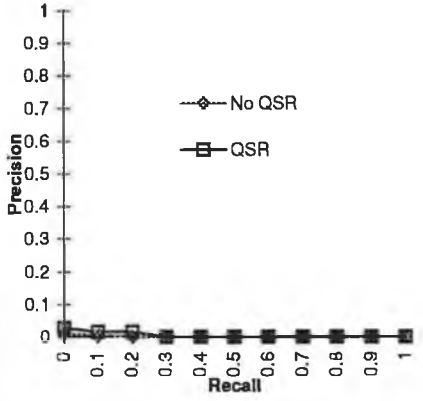


	No QSR	QSR	% Red.
Seconds:	14.6	6.3	56.85%
Doc. Acc:	434,766	50,000	88.50%



Percentage Reduction: 76.25%

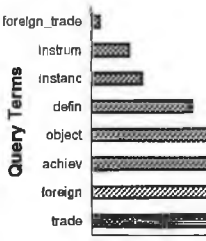
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	45	45
Rel_ret:	6	11
P. at 0.0	0.0098	0.0278
P. at 0.1	0.0066	0.0161
P. at 0.2	0	0.0158
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.001	0.0035
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0	0.02



	No QSR	QSR	% Red.
Seconds:	11.6	4	65.52%
Doc. Acc:	406,709	50,000	87.71%

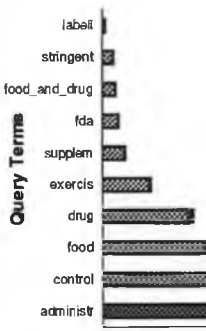
Query: 269

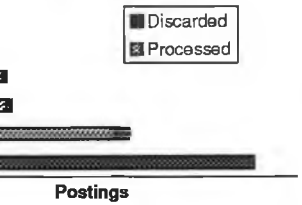
TERM	NP	QTT	PLT	Processed	Discarded
foreign_trade	2905	1	2905	2,905	0
instrum	12873	1	12844	12,844	29
instanc	17111	1	16882	16,882	229
defin	34166	1	33330	33,330	836
object	41181	1	39716	39,716	1,465
achiev	42981	1	40975	40,975	2,006
foreign	82835	1	78048	78,048	4,787
trade	124970	0	116360	0	124,970
	359,022			**224,700**	**134,322**



Query: 270

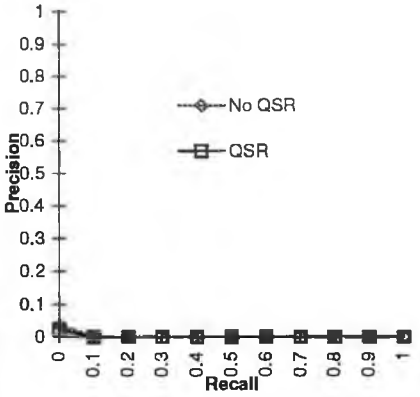
TERM	NP	QTT	PLT	Processed	Discarded
labell	927	1	927	927	0
stringent	3798	1	3798	3,798	0
food_and_drug	4508	1	4462	4,462	46
fda	5503	1	5392	5,392	111
supplem	7728	1	7496	7,496	232
exercis	16792	1	16120	16,120	672
drug	31471	1	29897	29,897	1,574
food	37715	1	35452	35,452	2,263
control	96879	1	90097	90,097	6,782
administr	124500	0	114539	0	124,500
	329,821			**193,641**	**136,180**



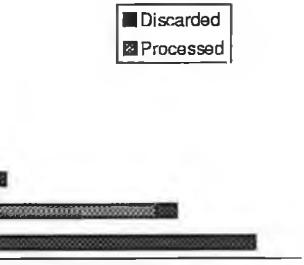


Percentage Reduction: 37.41%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	594	594
Rel_ret:	**20**	**12**
P. at 0.0	0.0367	0.0237
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0007**	**0.0003**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0.03	0.01

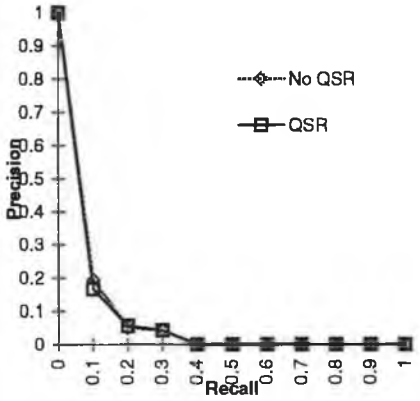


	No QSR	QSR	% Red.
Seconds:	9	5.1	43.33%
Doc. Acc:	291,815	50,000	82.87%



Percentage Reduction: 41.29%

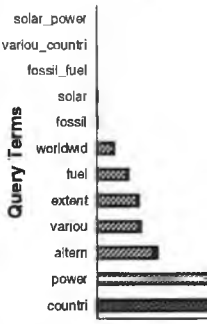
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	116	116
Rel_ret:	**42**	**42**
P. at 0.0	1	1
P. at 0.1	0.1935	0.1667
P. at 0.2	0.0507	0.0578
P. at 0.3	0.0425	0.0437
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0509**	**0.055**
P @ 10 D.	0.3	0.4
P @ 30 D.	0.267	0.267
P @ 100 D	0.13	0.15



	No QSR	QSR	% Red.
Seconds:	8.7	4.7	45.98%
Doc. Acc:	267,851	50,000	81.33%

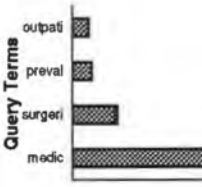
Query: 271

TERM	NP	QTT	PLT	Processed	Discarded
solar_power	123	1	123	123	0
variou_countri	143	1	143	143	0
fossil_fuel	552	1	548	548	4
solar	896	1	882	882	14
fossil	1043	1	1018	1,018	25
worldwid	10488	1	10155	10,155	333
fuel	19450	1	18672	18,672	778
extent	25721	1	24477	24,477	1,244
variou	26999	1	25469	25,469	1,530
altern	37184	1	34767	34,767	2,417
power	82890	1	76811	76,811	6,079
countri	237522	0	218124	0	237,522
	443,011			**193,065**	**249,946**



Query: 272

TERM	NP	QTT	PLT	Processed	Discarded
outpati	1111	1	1111	1,111	0
preval	1357	1	1340	1,340	17
surgeri	3121	1	3021	3,021	100
medic	27411	1	25985	25,985	1,426
	33,000			**31,457**	**1,543**

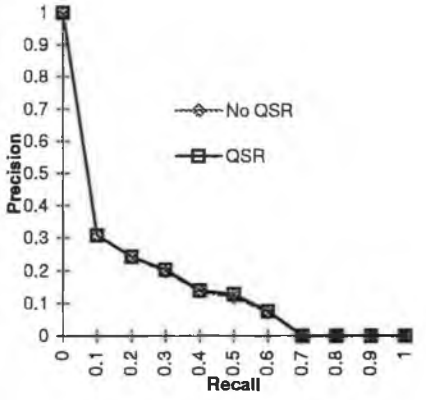


Discarded
Processed

Postings

Percentage Reduction: 56.42%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	86	86
Rel_ret:	**60**	**60**
P. at 0.0	1	1
P. at 0.1	0.3095	0.3095
P. at 0.2	0.2432	0.2432
P. at 0.3	0.1972	0.2031
P. at 0.4	0.1373	0.1401
P. at 0.5	0.1204	0.1276
P. at 0.6	0.0747	0.0765
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1423**	**0.1446**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.267	0.267
P @ 100 D.	0.21	0.21



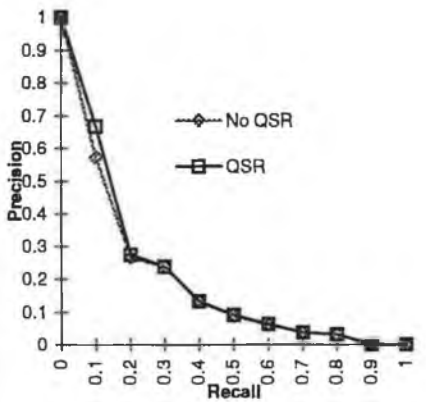
	No QSR	QSR	% Red.
Seconds:	10	5.1	49.00%
Doc. Acc:	377,593	50,000	86.76%

Discarded
Processed

Postings

Percentage Reduction: 4.68%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	36	36
Rel_ret:	**29**	**29**
P. at 0.0	1	1
P. at 0.1	0.5714	0.6667
P. at 0.2	0.2647	0.2727
P. at 0.3	0.2373	0.2373
P. at 0.4	0.1333	0.1333
P. at 0.5	0.09	0.0913
P. at 0.6	0.0615	0.0628
P. at 0.7	0.0378	0.0381
P. at 0.8	0.0316	0.0319
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1973**	**0.2026**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.233	0.233
P @ 100 D.	0.14	0.14



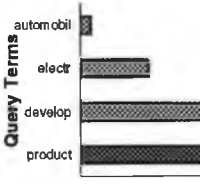
	No QSR	QSR	% Red.
Seconds:	2.5	2.4	4.00%
Doc. Acc:	31,062	29,610	4.67%

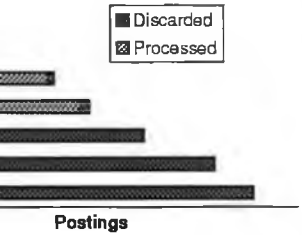
Query: 273

TERM	NP	QTT	PLT	Processed	Discarded
volcan	396	1	396	396	0
seismic	688	1	687	687	1
earthquak	2566	1	2537	2,537	29
determin_if	3484	1	3410	3,410	74
seek	49003	1	47483	47,483	1,520
signific	59445	1	57007	57,007	2,438
notic	75128	0	71296	0	75,128
activ	95674	0	89837	0	95,674
determin	107070	0	99468	0	107,070
	393,454			**111,520**	**281,934**

Query: 274

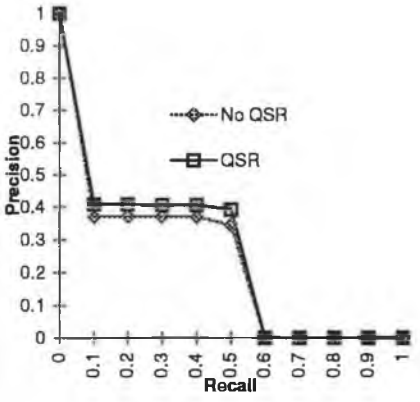
TERM	NP	QTT	PLT	Processed	Discarded
automobil	4874	1	4874	4,874	0
electr	32665	1	32273	32,273	392
develop	155682	1	150700	150,700	4,982
product	168048	0	159309	0	168,048
	361,269			**187,847**	**173,422**



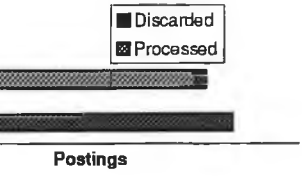


Percentage Reduction: 71.66%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	513	513
Rel_ret:	**289**	**303**
P. at 0.0	1	1
P. at 0.1	0.3712	0.4087
P. at 0.2	0.3712	0.4087
P. at 0.3	0.3712	0.4064
P. at 0.4	0.3712	0.4064
P. at 0.5	0.3456	0.3919
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1884**	**0.2178**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.3	0.3
P @ 100 D.	0.39	0.41

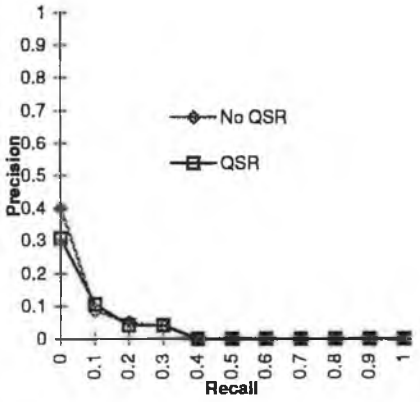


	No QSR	QSR	% Red.
Seconds:	9.9	4.5	54.55%
Doc. Acc:	303,018	50,000	83.50%



Percentage Reduction: 48.00%

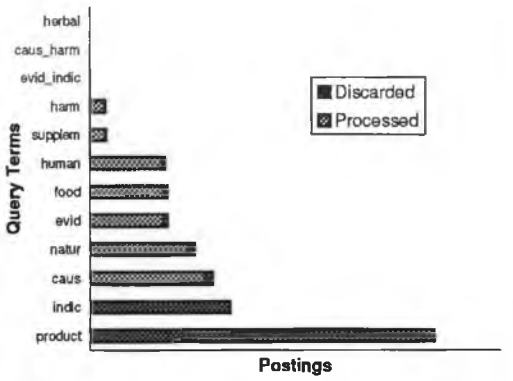
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	119	119
Rel_ret:	**39**	**39**
P. at 0.0	0.4	0.3077
P. at 0.1	0.087	0.1053
P. at 0.2	0.0502	0.0413
P. at 0.3	0.0392	0.0413
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0289**	**0.0279**
P @ 10 D.	0.2	0.2
P @ 30 D.	0.167	0.167
P @ 100 D.	0.1	0.09



	No QSR	QSR	% Red.
Seconds:	8.9	5.1	42.70%
Doc. Acc:	303,364	50,000	83.52%

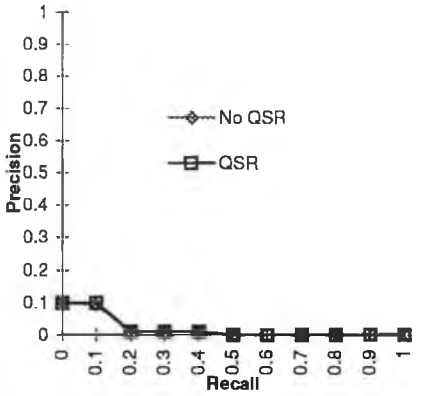
Query: 275

TERM	NP	QTT	PLT	Processed	Discarded
herbal	118	1	118	118	0
caus_harm	143	1	143	143	0
evid_indic	360	1	357	357	3
harm	7,340	1	7,229	7,229	111
supplem	7,728	1	7,547	7,547	181
human	36,323	1	35,172	35,172	1,151
food	37,715	1	36,206	36,206	1,509
evid	37,806	1	35,978	35,978	1,828
natur	51,136	1	48,238	48,238	2,898
caus	59,891	1	55,998	55,998	3,893
indic	68,326	0	63,315	0	68,326
product	168,048	0	154,324	0	168,048
	474,816			**226,868**	**247,948**



Percentage Reduction: 52.22%

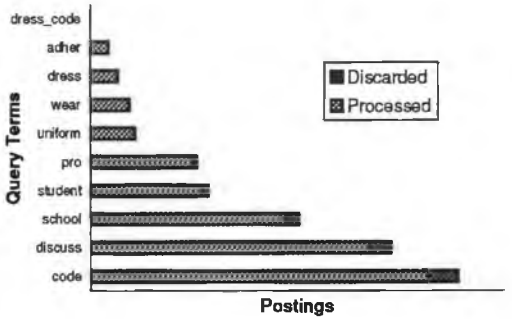
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	19	19
Rel_ret:	**9**	**8**
P. at 0.0	0.1	0.1
P. at 0.1	0.0952	0.1
P. at 0.2	0.0101	0.0109
P. at 0.3	0.01	0.0105
P. at 0.4	0.009	0.0093
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0136**	**0.0136**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.067	0.067
P @ 100 D	0.02	0.02



	No QSR	QSR	% Red.
Seconds:	10.3	5	51.46%
Doc. Acc:	370,730	50,000	86.51%

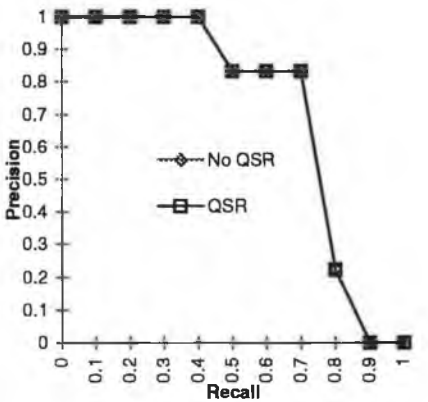
Query: 276

TERM	NP	QTT	PLT	Processed	Discarded
dress_code	121	1	121	121	0
adher	3,516	1	3,516	3,516	0
dress	5,352	1	5,298	5,298	54
wear	7,564	1	7,412	7,412	152
uniform	8,698	1	8,437	8,437	261
pro	20,664	1	19,837	19,837	827
student	22,747	1	21,609	21,609	1,138
school	40,281	1	37,864	37,864	2,417
discuss	58,190	1	54,116	54,116	4,074
code	71,043	1	65,359	65,359	5,684
	238,176			**223,569**	**14,607**



Percentage Reduction: 6.13%

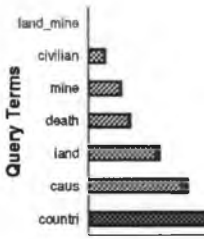
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	7	7
Rel_ret:	**6**	**6**
P. at 0.0	1	1
P. at 0.1	1	1
P. at 0.2	1	1
P. at 0.3	1	1
P. at 0.4	1	1
P. at 0.5	0.8333	0.8333
P. at 0.6	0.8333	0.8333
P. at 0.7	0.8333	0.8333
P. at 0.8	0.2222	0.2222
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.6937**	**0.6937**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.2	0.2
P @ 100 D	0.06	0.06



	No QSR	QSR	% Red.
Seconds:	7.6	5.3	30.26%
Doc. Acc:	204,823	50,000	75.59%

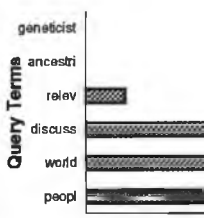
Query: 277

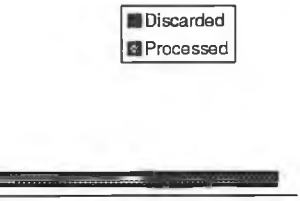
TERM	NP	QTT	PLT	Processed	Discarded
land_mine	154	1	154	154	0
civilian	10,016	1	9,978	9,978	38
mine	19,870	1	19,547	19,547	323
death	25,116	1	24,393	24,393	723
land	42,925	1	41,154	41,154	1,771
caus	59,891	1	56,671	56,671	3,220
countri	237,522	0	221,786	0	237,522
	395,494			**151,897**	**243,597**



Query: 278

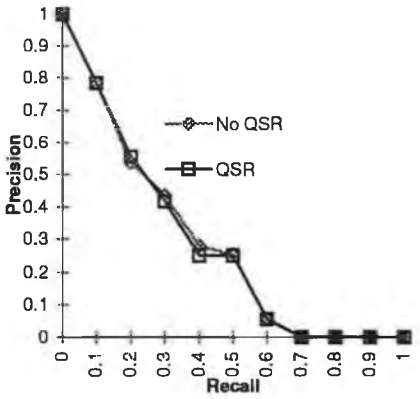
TERM	NP	QTT	PLT	Processed	Discarded
geneticist	95	1	95	95	0
ancestri	292	1	290	290	2
relev	15,178	1	14,874	14,874	304
discuss	58,190	1	56,194	56,194	1,996
world	96,436	1	91,751	91,751	4,685
peopl	147,661	0	138,379	0	147,661
	318,130			**163,204**	**154,648**



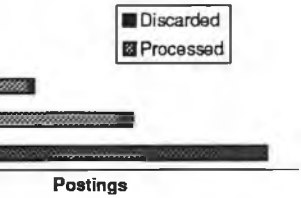


Percentage Reduction: 61.59%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	74	74
Rel_ret:	48	48
P. at 0.0	1	1
P. at 0.1	0.7857	0.7857
P. at 0.2	0.5357	0.5556
P. at 0.3	0.4364	0.4182
P. at 0.4	0.2807	0.2517
P. at 0.5	0.2533	0.2517
P. at 0.6	0.0578	0.0549
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.2625	0.2542
P @ 10 D.	0.7	0.7
P @ 30 D.	0.5	0.5
P @ 100 D.	0.28	0.25

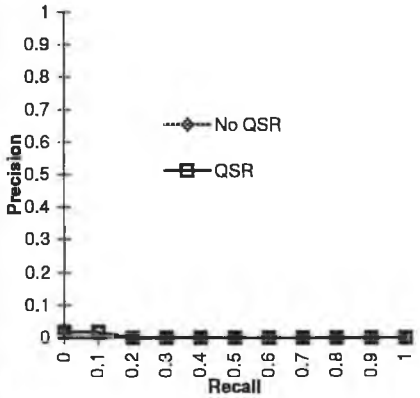


	No QSR	QSR	% Red.
Seconds:	11.5	4.4	61.74%
Doc. Acc:	344,358	50,000	85.48%



Percentage Reduction: 48.61%

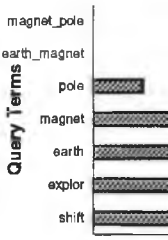
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	7	7
Rel_ret:	1	1
P. at 0.0	0.0182	0.0175
P. at 0.1	0.0182	0.0175
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	0.0026	0.0025
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0.01	0.01



	No QSR	QSR	% Red.
Seconds:	8.3	4.2	49.40%
Doc. Acc:	273,950	50,000	81.75%

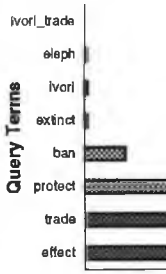
Query: 279

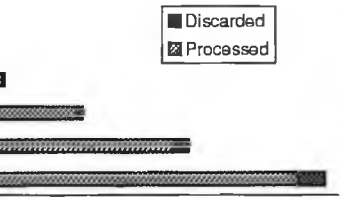
TERM	NP	QTT	PLT	Processed	Discarded
magnet_pole	9	1	9	9	0
earth_magnet	29	1	28	28	1
pole	2,169	1	2,133	2,133	36
magnet	3,454	1	3,354	3,354	100
earth	6,859	1	6,576	6,576	283
explor	11,482	1	10,864	10,864	618
shift	17.400	1	16,247	16,247	1,153
	41,681			**39,211**	**2,191**



Query: 280

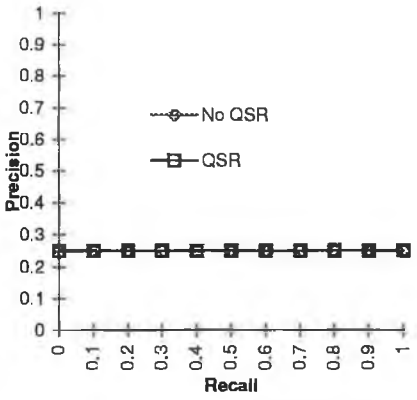
TERM	NP	QTT	PLT	Processed	Discarded
ivori_trade	36	1	36	36	0
eleph	941	1	938	938	3
ivori	1,092	1	1,077	1,077	15
extinct	1,178	1	1,149	1,149	29
ban	13,771	1	13,281	13,281	490
protect	72,331	1	68,955	68,955	3,376
trade	124,970	0	117,749	0	124,970
effect	135,381	0	126,054	0	135,381
	349,980			**85,436**	**264,264**



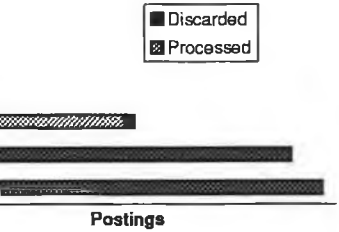


Percentage Reduction: 5.26%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	2	2
Rel_ret:	**2**	**2**
P. at 0.0	0.25	0.25
P. at 0.1	0.25	0.25
P. at 0.2	0.25	0.25
P. at 0.3	0.25	0.25
P. at 0.4	0.25	0.25
P. at 0.5	0.25	0.25
P. at 0.6	0.25	0.25
P. at 0.7	0.25	0.25
P. at 0.8	0.25	0.25
P. at 0.9	0.25	0.25
P. at 1.0	0.25	0.25
Av. P	**0.2083**	**0.2083**
P @ 10 D.	0.2	0.2
P @ 30 D.	0.067	0.067
P @ 100 D	0.02	0.02

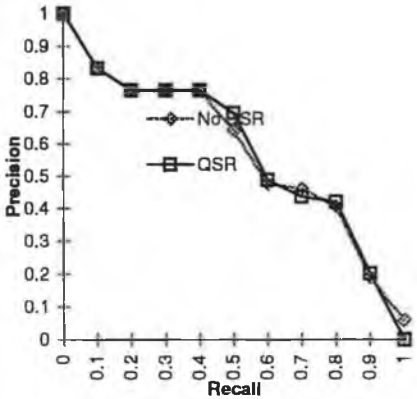


	No QSR	QSR	% Red.
Seconds:	2.9	2.8	3.45%
Doc. Acc:	40,143	38,026	5.27%



Percentage Reduction: 75.51%

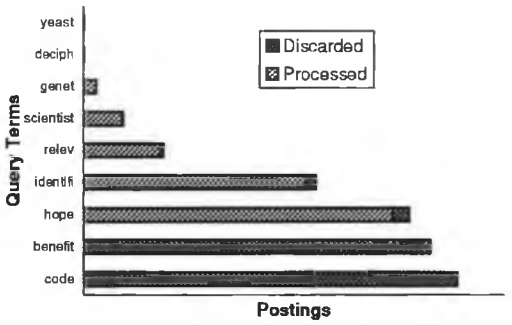
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	32	32
Rel_ret:	**32**	**31**
P. at 0.0	1	1
P. at 0.1	0.8333	0.8333
P. at 0.2	0.7647	0.7647
P. at 0.3	0.7647	0.7647
P. at 0.4	0.7647	0.7647
P. at 0.5	0.64	0.6957
P. at 0.6	0.4762	0.4878
P. at 0.7	0.46	0.4364
P. at 0.8	0.4062	0.4194
P. at 0.9	0.1883	0.2014
P. at 1.0	0.0578	0
Av. P	**0.5604**	**0.5653**
P @ 10 D.	0.6	0.6
P @ 30 D.	0.533	0.567
P @ 100 D	0.27	0.27



	No QSR	QSR	% Red.
Seconds:	9	3.8	57.78%
Doc. Acc:	301,021	50,000	83.39%

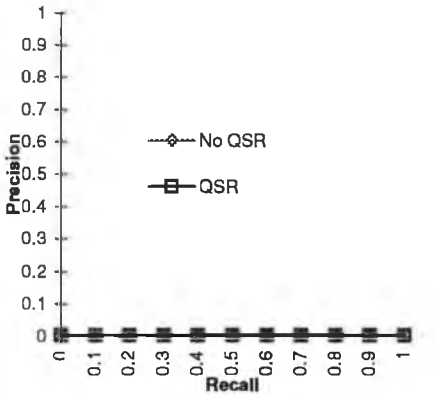
Query: 281

TERM	NP	QTT	PLT	Processed	Discarded
yeast	203	1	203	203	0
deciph	247	1	246	246	1
genet	2,429	1	2,402	2,402	27
scientist	7,579	1	7,419	7,419	160
relev	15,178	1	14,707	14,707	471
identifi	44,281	1	42,465	42,465	1,816
hope	61,932	1	58,773	58,773	3,159
benefit	65,942	0	61,919	0	65,942
code	71,043	0	65,998	0	71,043
	269,115			**126,215**	**142,619**



Percentage Reduction: 53.00%

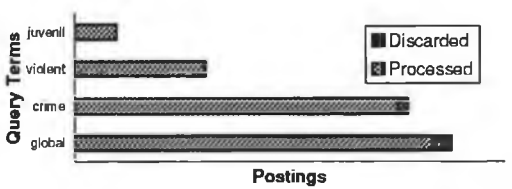
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	1	1
Rel_ret:	**1**	**1**
P. at 0.0	0.0042	0.0046
P. at 0.1	0.0042	0.0046
P. at 0.2	0.0042	0.0046
P. at 0.3	0.0042	0.0046
P. at 0.4	0.0042	0.0046
P. at 0.5	0.0042	0.0046
P. at 0.6	0.0042	0.0046
P. at 0.7	0.0042	0.0046
P. at 0.8	0.0042	0.0046
P. at 0.9	0.0042	0.0046
P. at 1.0	0.0042	0.0046
Av. P	**0.0042**	**0.0046**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0	0



	No QSR	QSR	% Red.
Seconds:	7.8	4.2	46.15%
Doc. Acc:	238,818	50,000	79.06%

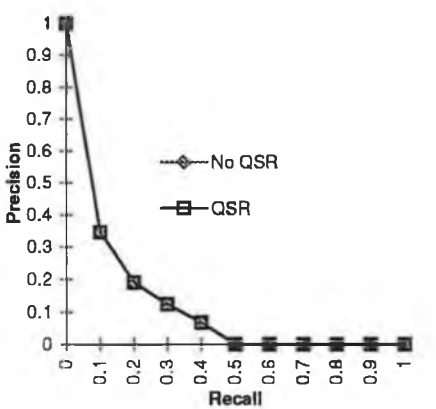
Query: 282

TERM	NP	QTT	PLT	Processed	Discarded
juvenil	1,935	1	1,935	1,935	0
violent	6,088	1	6,014	6,014	74
crime	15,512	1	15,015	15,015	497
global	17,528	1	16,616	16,616	912
	41,345			**39,580**	**1,483**



Percentage Reduction: 3.59%

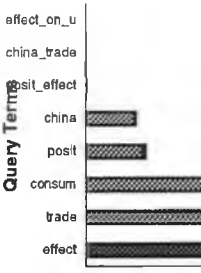
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	131	131
Rel_ret:	**57**	**57**
P. at 0.0	1	1
P. at 0.1	0.3455	0.3455
P. at 0.2	0.1895	0.1908
P. at 0.3	0.1239	0.1239
P. at 0.4	0.0675	0.0676
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0998**	**0.1**
P @ 10 D.	0.5	0.5
P @ 30 D.	0.4	0.4
P @ 100 D	0.22	0.22



	No QSR	QSR	% Red.
Seconds:	2.9	2.8	3.45%
Doc. Acc:	38,157	36,795	3.57%

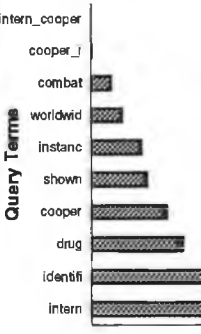
Query: 283

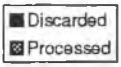
TERM	NP	QTT	PLT	Processed	Discarded
effect_on_u	133	1	133	133	0
china_trade	177	1	176	176	1
posit_effect	488	1	481	481	7
china	16,977	1	16,562	16,562	415
posit	20,109	1	19,394	19,394	715
consum	48,666	1	46,394	46,394	2,272
trade	124,970	1	117,749	117,749	7,221
effect	135,381	0	126.054	0	135,381
	347,184			**200,889**	**146,012**



Query: 284

TERM	NP	QTT	PLT	Processed	Discarded
intern_cooper	297	1	297	297	0
cooper_i	333	1	333	333	0
combat	6,830	1	6,761	6,761	69
worldwid	10,488	1	10,278	10,278	210
instanc	17,111	1	16,597	16,597	514
shown	19,137	1	18,371	18,371	766
cooper	25,912	1	24,616	24,616	1,296
drug	31,471	1	29,582	29,582	1,889
identifi	44,281	1	41,181	41,181	3,100
intern	129,101	1	118,772	118,772	10,329
	284,961			**266,788**	**18,173**

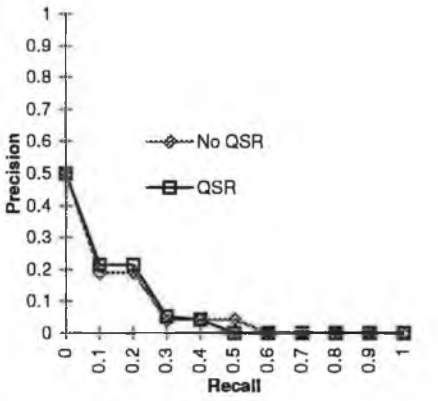




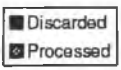
Postings

Percentage Reduction: 42.06%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	84	84
Rel_ret:	**42**	**41**
P. at 0.0	0.5	0.5
P. at 0.1	0.1895	0.2143
P. at 0.2	0.1895	0.2143
P. at 0.3	0.043	0.0522
P. at 0.4	0.043	0.0421
P. at 0.5	0.0426	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0633**	**0.0677**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.133	0.133
P @ 100 D.	0.18	0.19



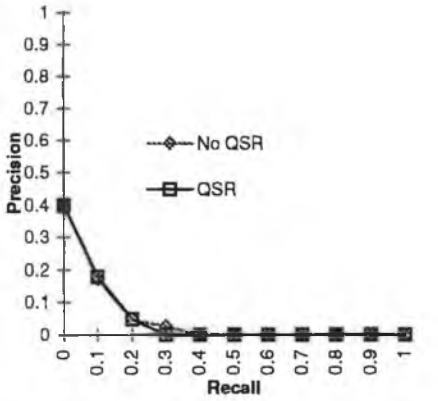
	No QSR	QSR	% Red.
Seconds:	10.3	4.8	53.40%
Doc. Acc:	293,952	50,000	82.99%



Postings

Percentage Reduction: 6.38%

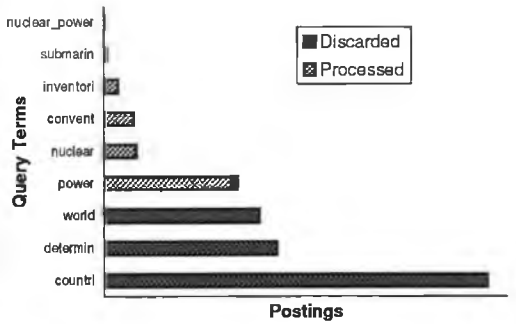
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	70	70
Rel_ret:	**24**	**20**
P. at 0.0	0.4	0.4
P. at 0.1	0.1731	0.18
P. at 0.2	0.0473	0.0491
P. at 0.3	0.0264	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0344**	**0.0336**
P @ 10 D.	0.2	0.2
P @ 30 D.	0.167	0.167
P @ 100 D.	0.1	0.1



	No QSR	QSR	% Red.
Seconds:	8.3	5.7	31.33%
Doc. Acc:	250,223	50,000	80.02%

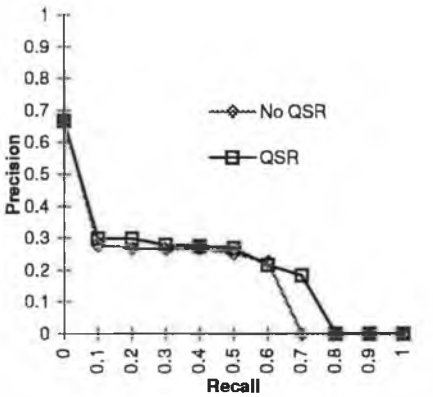
Query: 285

TERM	NP	QTT	PLT	Processed	Discarded
nuclear_power	1,616	1	1,616	1,616	0
submarin	2,168	1	2,165	2,165	3
inventori	8,927	1	8,828	8,828	99
convent	18,647	1	18,255	18,255	392
nuclear	20,049	1	19,427	19,427	622
power	82,890	1	79,491	79,491	3,399
world	96,436	0	91,517	0	96,436
determin	107,070	0	100,538	0	107,070
countri	237,522	0	220,657	0	237,522
	575,610			**129,782**	**445,543**



Percentage Reduction: 77.40%

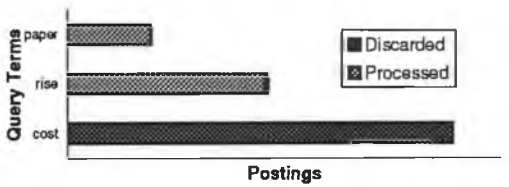
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	261	261
Rel_ret:	**178**	**183**
P. at 0.0	0.6667	0.6667
P. at 0.1	0.2774	0.2978
P. at 0.2	0.2672	0.2978
P. at 0.3	0.2656	0.2781
P. at 0.4	0.2655	0.2751
P. at 0.5	0.2513	0.269
P. at 0.6	0.2256	0.2164
P. at 0.7	0	0.1834
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.1756**	**0.189**
P @ 10 D.	0.4	0.4
P @ 30 D.	0.333	0.367
P @ 100 D.	0.26	0.26



	No QSR	QSR	% Red.
Seconds:	11.2	4	64.29%
Doc. Acc:	453,323	50,000	88.97%

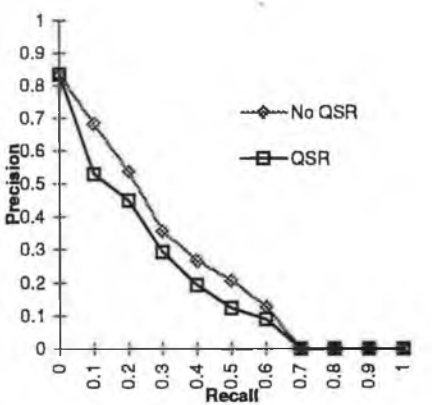
Query: 286

TERM	NP	QTT	PLT	Processed	Discarded
paper	30,918	1	30,918	30,918	0
rise	74,108	1	72,811	72,811	1,297
cost	142,167	0	136,124	0	142,167
	247,479			**103,729**	**143,464**



Percentage Reduction: 57.97%

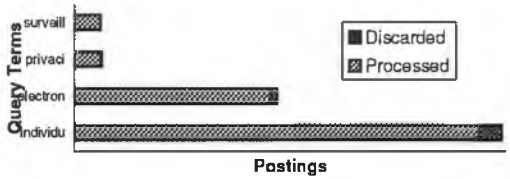
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	142	142
Rel_ret:	**90**	**88**
P. at 0.0	0.8333	0.8333
P. at 0.1	0.6818	0.5294
P. at 0.2	0.537	0.4493
P. at 0.3	0.3554	0.2924
P. at 0.4	0.2667	0.1942
P. at 0.5	0.208	0.1248
P. at 0.6	0.1278	0.0908
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.2485**	**0.1934**
P @ 10 D.	0.7	0.6
P @ 30 D.	0.633	0.467
P @ 100 D.	0.37	0.34



	No QSR	QSR	% Red.
Seconds:	7.8	3.8	51.28%
Doc. Acc:	222,254	50,000	77.50%

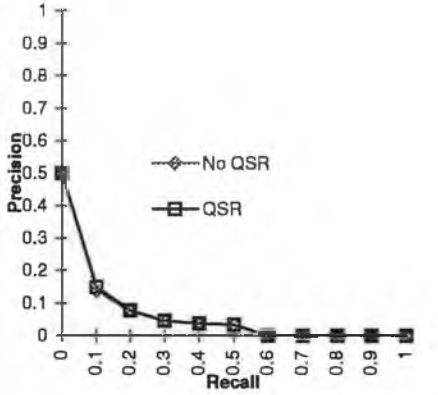
Query: 287

TERM	NP	QTT	PLT	Processed	Discarded
surveill	4,116	1	4,116	4,116	0
privaci	4,323	1	4,271	4,271	52
electron	32,814	1	31,763	31,763	1,051
individu	69,284	1	65,681	65,681	3,603
	110,824			**105,831**	**4,706**



Percentage Reduction: 4.25%

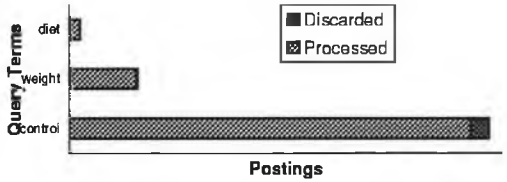
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	40	40
Rel_ret:	**23**	**23**
P. at 0.0	0.5	0.5
P. at 0.1	0.1379	0.1481
P. at 0.2	0.0748	0.0769
P. at 0.3	0.0453	0.0458
P. at 0.4	0.0377	0.0383
P. at 0.5	0.0334	0.0338
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0475**	**0.0483**
P @ 10 D.	0.2	0.2
P @ 30 D.	0.133	0.133
P @ 100 D.	0.07	0.07



	No QSR	QSR	% Red.
Seconds:	5.2	3.6	30.77%
Doc. Acc:	105,784	50,000	52.73%

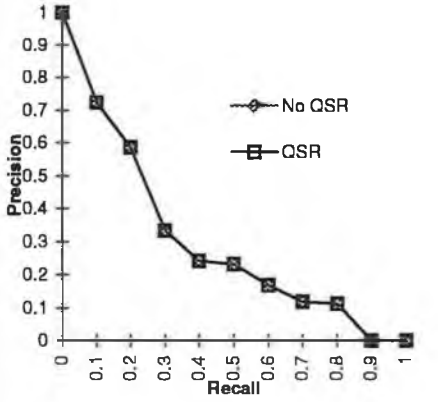
Query: 288

TERM	NP	QTT	PLT	Processed	Discarded
diet	2,360	1	2,360	2,360	0
weight	15,609	1	15,335	15,335	274
control	96,879	1	92,761	92,761	4,118
	115,136			**110,456**	**4,392**



Percentage Reduction: 3.81%

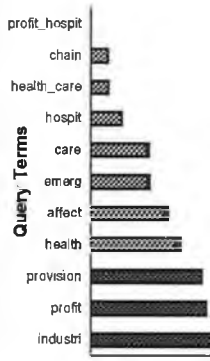
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	92	92
Rel_ret:	**77**	**77**
P. at 0.0	1	1
P. at 0.1	0.7222	0.7222
P. at 0.2	0.5854	0.5854
P. at 0.3	0.3333	0.3333
P. at 0.4	0.2418	0.2418
P. at 0.5	0.2315	0.2315
P. at 0.6	0.1687	0.169
P. at 0.7	0.1185	0.1189
P. at 0.8	0.1128	0.1131
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.3134**	**0.3122**
P @ 10 D.	0.8	0.8
P @ 30 D.	0.567	0.567
P @ 100 D.	0.3	0.3



	No QSR	QSR	% Red.
Seconds:	5.2	3.8	26.92%
Doc. Acc:	112,573	50,000	55.58%

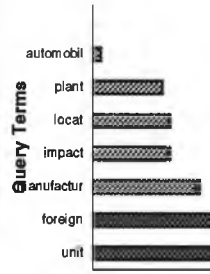
Query: 289

TERM	NP	QTT	PLT	Processed	Discarded
profit_hospit	54	1	54	54	0
chain	12,037	1	12,037	12,037	0
health_care	12,318	1	12,217	12,217	101
hospit	22,294	1	21,908	21,908	386
care	41,317	1	40,227	40,227	1,090
emerg	42,098	1	40,605	40,605	1,493
affect	55,078	1	52,624	52,624	2,454
health	63,917	1	60,488	60,488	3,429
provision	78,957	0	74,004	0	78,957
profit	82,449	0	76,527	0	82,449
industri	259,266	0	238,289	0	259,266
	669,785			**240,160**	**429,625**



Query: 290

TERM	NP	QTT	PLT	Processed	Discarded
automobil	4,874	1	4,874	4,874	0
plant	41,960	1	41,802	41,802	158
locat	46,533	1	45,776	45,776	757
impact	46,830	1	45,483	45,483	1,347
manufactur	64,464	1	61,804	61,804	2,660
foreign	82,835	0	78,382	0	82,835
unit	236,217	0	220,567	0	236,217
	524,003			**199,739**	**323,974**

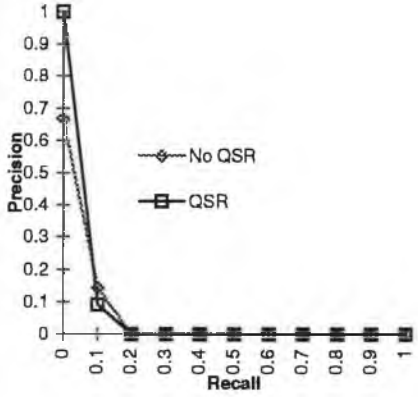


Discarded
Processed

Postings

Percentage Reduction: 64.14%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	141	141
Rel_ret:	**26**	**26**
P. at 0.0	0.6667	1
P. at 0.1	0.1442	0.0914
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0333**	**0.0373**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.2	0.167
P @ 100 D	0.14	0.13



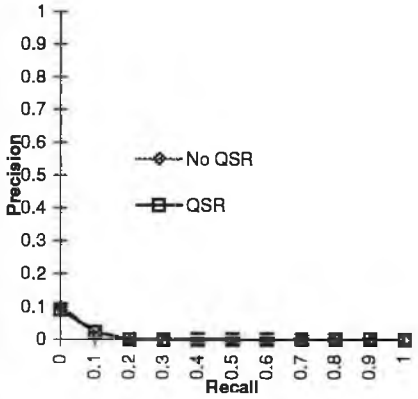
	No QSR	QSR	% Red.
Seconds:	12.8	5.4	57.81%
Doc. Acc:	481,713	50,000	89.62%

Discarded
Processed

Postings

Percentage Reduction: 61.63%

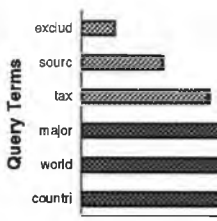
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	119	119
Rel_ret:	**21**	**20**
P. at 0.0	0.1	0.0909
P. at 0.1	0.0238	0.021
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0051**	**0.0046**
P @ 10 D.	0.1	0
P @ 30 D.	0.067	0.067
P @ 100 D.	0.02	0.02



	No QSR	QSR	% Red.
Seconds:	11.6	4.8	58.62%
Doc. Acc:	414,176	50,000	87.93%

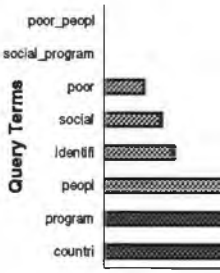
Query: 291

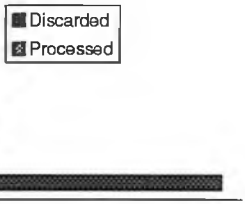
TERM	NP	QTT	PLT	Processed	Discarded
exclud	19,609	1	19,609	19,609	0
sourc	48,725	1	48,446	48,446	279
tax	76,398	1	74,870	74,870	1,528
major	95,810	0	92,525	0	95,810
world	96,436	0	91,751	0	96,436
countri	237,522	0	222,592	0	237,522
	574,791			**142,925**	**431,575**



Query: 292

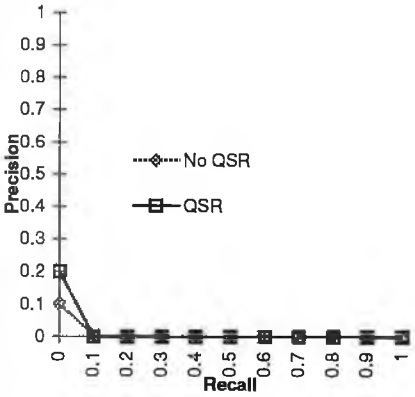
TERM	NP	QTT	PLT	Processed	Discarded
poor_peopl	551	1	551	551	0
social_program	584	1	582	582	2
poor	25,388	1	25,049	25,049	339
social	36,468	1	35,576	35,576	892
identifi	44,281	1	42,706	42,706	1,575
peopl	147,661	1	140,770	140,770	6,891
program	151,200	0	142,464	0	151,200
countri	237,522	0	221,159	0	237,522
	643,947			**245,234**	**398,421**





Percentage Reduction: 75.08%

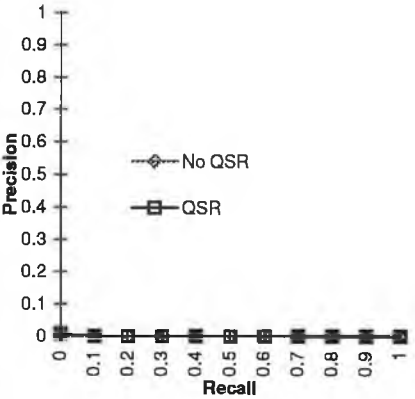
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	407	407
Rel_ret:	**30**	**28**
P. at 0.0	0.102	0.2
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0032**	**0.004**
P @ 10 D.	0	0.1
P @ 30 D.	0.067	0.133
P @ 100 D	0.05	0.06



	No QSR	QSR	% Red.
Seconds:	13.7	4.1	70.07%
Doc. Acc:	434,140	50,000	88.48%

Percentage Reduction: 61.87%

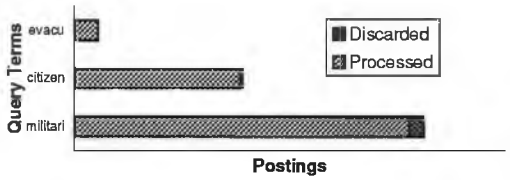
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	59	59
Rel_ret:	**3**	**3**
P. at 0.0	0.0078	0.0081
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0003**	**0.0003**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D	0	0



	No QSR	QSR	% Red.
Seconds:	12.9	5.2	59.69%
Doc. Acc:	493,289	50,000	89.86%

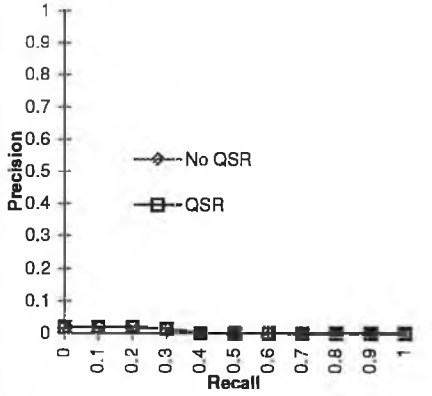
Query: 293

TERM	NP	QTT	PLT	Processed	Discarded
evacu	2,718	1	2,718	2,718	0
citizen	19,518	1	19,176	19,176	342
militari	40,513	1	38,791	38,791	1,722
	63,042			**60,685**	**2,064**



Percentage Reduction: 3.27%

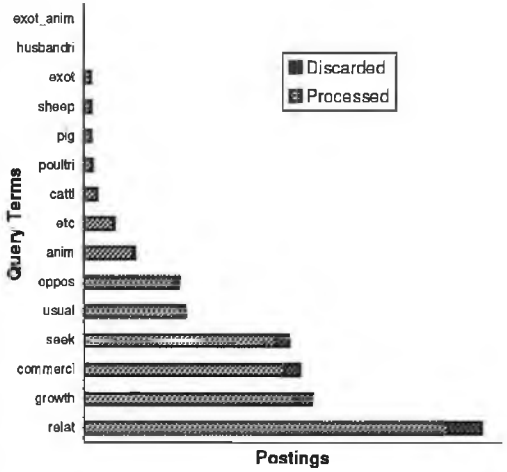
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	41	41
Rel_ret:	**13**	**13**
P. at 0.0	0.0197	0.0198
P. at 0.1	0.0197	0.0198
P. at 0.2	0.0197	0.0198
P. at 0.3	0.0135	0.0135
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0042**	**0.0042**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0.01	0.01



	No QSR	QSR	% Red.
Seconds:	3.9	3.5	10.26%
Doc. Acc:	60,516	50,000	17.38%

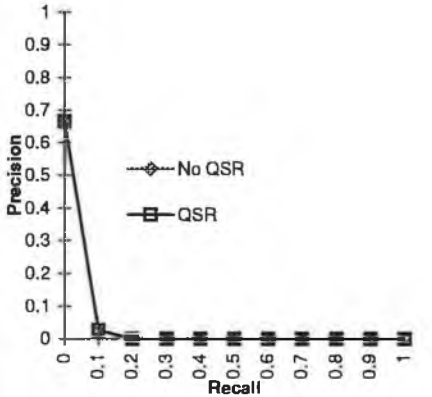
Query: 294

TERM	NP	QTT	PLT	Processed	Discarded
exot_anim	65	1	65	65	0
husbandri	114	1	114	114	0
exot	1,577	1	1,571	1,571	6
sheep	1,617	1	1,600	1,600	17
pig	1,638	1	1,610	1,610	28
poultri	1,918	1	1,873	1,873	45
cattl	3,010	1	2,919	2,919	91
etc	7,173	1	6,909	6,909	264
anim	12,100	1	11,575	11,575	525
oppos	22,632	1	21,500	21,500	1,132
usual	24,156	1	22,787	22,787	1,369
seek	49,003	1	45,899	45,899	3,104
commerci	51,581	1	47,970	47,970	3,611
growth	54,553	1	50,370	50,370	4,183
relat	95,201	1	87,267	87,267	7,934
	326,338			**304,029**	**22,309**



Percentage Reduction: 6.84%

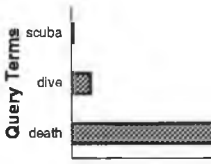
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	160	160
Rel_ret:	**21**	**21**
P. at 0.0	0.6667	0.6667
P. at 0.1	0.0254	0.0262
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0176**	**0.0177**
P @ 10 D.	0.3	0.3
P @ 30 D.	0.133	0.133
P @ 100 D.	0.08	0.08



	No QSR	QSR	% Red.
Seconds:	10.1	7.5	25.74%
Doc. Acc:	278,401	50,000	82.04%

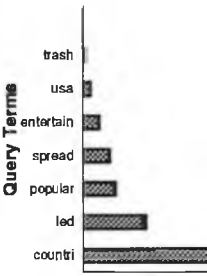
Query: 295

TERM	NP	QTT	PLT	Processed	Discarded
scuba	127	1	127	127	0
dive	1,326	1	1,302	1,302	24
death	25,116	1	24,048	24,048	1,068
	26,864			**25,477**	**1,092**



Query: 296

TERM	NP	QTT	PLT	Processed	Discarded
trash	1,370	1	1,370	1,370	0
usa	4,587	1	4,569	4,569	18
entertain	9,973	1	9,810	9,810	163
spread	16,276	1	15,808	15,808	468
popular	19,650	1	18,839	18,839	811
led	37,955	1	35,914	35,914	2,041
countri	237,522	1	221,786	221,786	15,736
	327,629			**308,096**	**19,237**

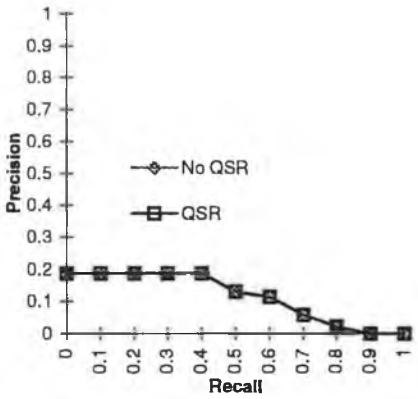


Discarded
Processed

Postings

Percentage Reduction: 4.06%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	15	15
Rel_ret:	**12**	**12**
P. at 0.0	0.1875	0.1875
P. at 0.1	0.1875	0.1875
P. at 0.2	0.1875	0.1875
P. at 0.3	0.1875	0.1875
P. at 0.4	0.1875	0.1875
P. at 0.5	0.129	0.1311
P. at 0.6	0.1125	0.1154
P. at 0.7	0.0576	0.0579
P. at 0.8	0.0218	0.0223
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0887**	**0.0893**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.133	0.133
P @ 100 D	0.09	0.09



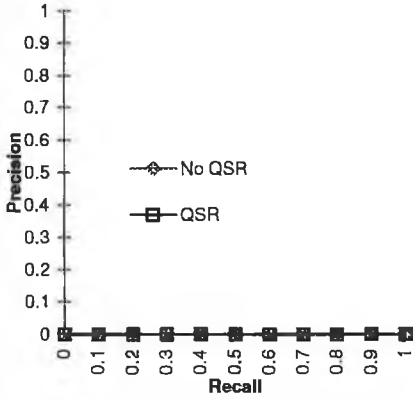
	No QSR	QSR	% Red.
Seconds:	2.6	2.5	3.85%
Doc. Acc:	26,432	25,347	4.10%

Discarded
Processed

Postings

Percentage Reduction: 5.87%

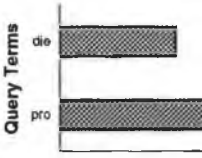
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	1	1
Rel_ret:	**0**	**0**
P. at 0.0	0	0
P. at 0.1	0	0
P. at 0.2	0	0
P. at 0.3	0	0
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0**	**0**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0	0



	No QSR	QSR	% Red.
Seconds:	11.8	7.2	38.98%
Doc. Acc:	294,735	50,000	83.04%

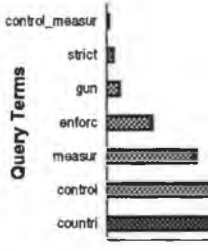
Query: 297

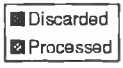
TERM	NP	QTT	PLT	Processed	Discarded
die	6,538	1	6,538	6,538	0
pro	20,664	1	20,112	20,112	552
	27,499			**26,650**	**552**



Query: 298

TERM	NP	QTT	PLT	Processed	Discarded
control_measur	1,118	1	1,118	1,118	0
strict	4,052	1	4,036	4,036	16
gun	7,919	1	7,790	7,790	129
enforc	28,673	1	27,848	27,848	825
measur	56,018	1	53,707	53,707	2,311
control	96,879	1	91,671	91,671	5,208
countri	237,522	0	221,786	0	237,522
	432,479			**186,170**	**246,011**

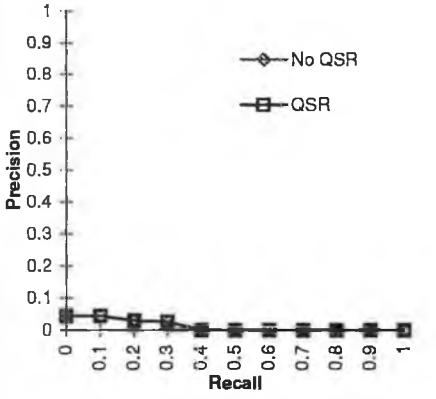




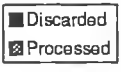
Postings

Percentage Reduction: 2.01%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	86	86
Rel_ret:	**26**	**26**
P. at 0.0	0.044	0.044
P. at 0.1	0.044	0.044
P. at 0.2	0.0298	0.0298
P. at 0.3	0.0267	0.0267
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0093**	**0.0093**
P @ 10 D.	0	0
P @ 30 D.	0	0
P @ 100 D.	0.01	0.01



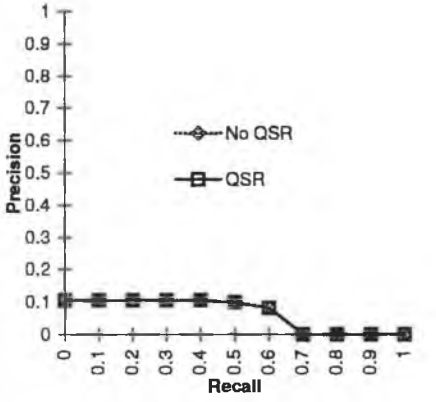
	No QSR	QSR	% Red.
Seconds:	2.1	2.1	0.00%
Doc. Acc:	27,004	26,459	2.02%



Postings

Percentage Reduction: 56.88%

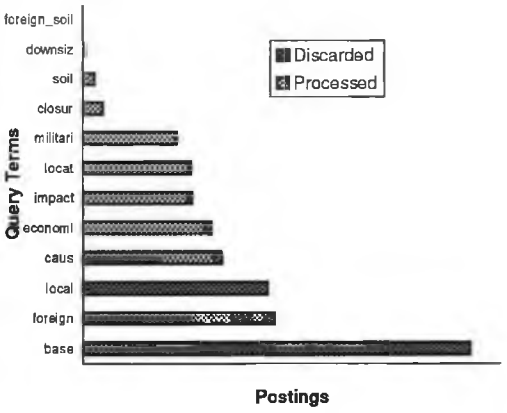
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	91	91
Rel_ret:	**60**	**62**
P. at 0.0	0.1061	0.1056
P. at 0.1	0.1061	0.1056
P. at 0.2	0.1061	0.1056
P. at 0.3	0.1061	0.1056
P. at 0.4	0.1061	0.1056
P. at 0.5	0.0991	0.1013
P. at 0.6	0.0825	0.0828
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0544**	**0.056**
P @ 10 D.	0	0
P @ 30 D.	0.067	0.033
P @ 100 D.	0.07	0.07



	No QSR	QSR	% Red.
Seconds:	9.7	4.7	51.55%
Doc. Acc:	371,754	50,000	86.55%

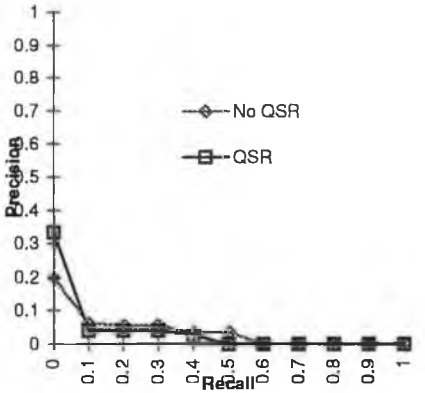
Query: 299

TERM	NP	QTT	PLT	Processed	Discarded
foreign_soil	46	1	46	46	0
downsiz	1,157	1	1,157	1,157	0
soil	4,928	1	4,895	4,895	33
closur	8,642	1	8,512	8,512	130
militari	40,513	1	39,567	39,567	946
locat	46,533	1	45,059	45,059	1,474
impact	46,830	1	44,956	44,956	1,874
economi	55,130	1	52,465	52,465	2,665
caus	59,891	1	56,497	56,497	3,394
local	79,412	0	74,250	0	79,412
foreign	82,835	0	76,760	0	82,835
base	166,964	0	153,328	0	166,964
	592,835			**253,108**	**339,727**



Percentage Reduction: 57.31%

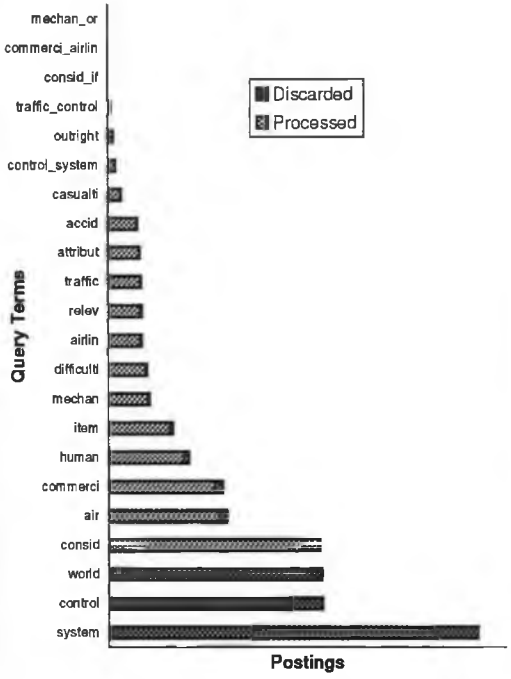
	No QSR	QSR
Retrieved:	1000	1000
Relevant:	62	62
Rel_ret:	**33**	**27**
P. at 0.0	0.2	0.3333
P. at 0.1	0.0625	0.04
P. at 0.2	0.0561	0.04
P. at 0.3	0.0561	0.04
P. at 0.4	0.0354	0.0278
P. at 0.5	0.0354	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0309**	**0.0246**
P @ 10 D.	0.1	0.1
P @ 30 D.	0.1	0.1
P @ 100 D.	0.06	0.05



	No QSR	QSR	% Red.
Seconds:	12	5.7	52.50%
Doc. Acc:	432,024	50,000	88.43%

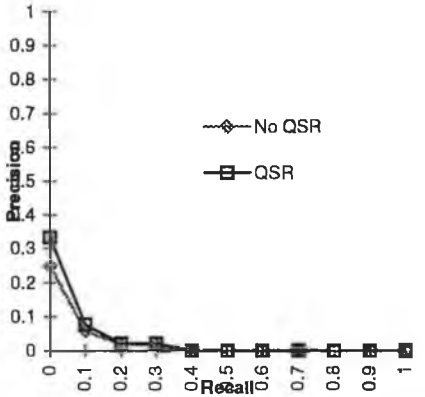
Query: 300

TERM	NP	QTT	PLT	Processed	Discarded
mechan_or	278	1	278	278	0
commerci_airlin	337	1	337	337	0
consid_if	400	1	400	400	0
traffic_control	1,178	1	1,173	1,173	5
outright	2,353	1	2,332	2,332	21
control_system	3,330	1	3,285	3,285	45
casualti	5,894	1	5,787	5,787	107
accid	12,941	1	12,645	12,645	296
attribut	14,214	1	13,821	13,821	393
traffic	14,700	1	14,224	14,224	476
relev	15,178	1	14,614	14,614	564
airlin	15,206	1	14,568	14,568	638
difficulti	17,263	1	16,457	16,457	806
mechan	18,532	1	17,578	17,578	954
item	29,027	1	27,395	27,395	1,632
human	36,323	1	34,109	34,109	2,214
commerci	51,581	1	48,191	48,191	3,390
air	53,654	1	49,872	49,872	3,782
consid	95,644	1	88,447	88,447	7,197
world	96,436	0	88,721	0	96,436
control	96,879	0	88,667	0	96,879
system	167,317	0	152,338	0	167,317
	748,665			**365,513**	**383,152**



Percentage Reduction: 51.18%

	No QSR	QSR
Retrieved:	1000	1000
Relevant:	44	44
Rel_ret:	**15**	**15**
P. at 0.0	0.25	0.3333
P. at 0.1	0.0568	0.0758
P. at 0.2	0.0186	0.0219
P. at 0.3	0.0165	0.0203
P. at 0.4	0	0
P. at 0.5	0	0
P. at 0.6	0	0
P. at 0.7	0	0
P. at 0.8	0	0
P. at 0.9	0	0
P. at 1.0	0	0
Av. P	**0.0227**	**0.0294**
P @ 10 D.	0.1	0.3
P @ 30 D.	0.1	0.1
P @ 100 D.	0.05	0.05



	No QSR	QSR	% Red.
Seconds:	14.2	7.5	47.18%
Doc. Acc:	499,742	50,000	89.99%

Appendix C

Sample Stopword List.

A	Cannot	Into	Our	Thus
About	Co	Is	Ours	To
Above	Could	It	Ourselves	Together
Across	Down	Its	Out	Too
After	During	Itself	Over	Toward
Afterwards	Each	Last	Own	Towards
Again	Eg	Latter	Per	Under
Against	Either	Latterly	Perhaps	Until
All	Else	Least	Rather	Up
Almost	Elsewhere	Less	Same	Upon
Alone	Enough	Ltd	Seem	Us
Along	Etc	Many	Seemed	Very
Already	Even	May	Seeming	Via
Also	Ever	Me	Seems	Was
Although	Every	Meanwhile	Several	We
Always	Everyone	Might	She	Well
Among	Everything	More	Should	Were
Amongst	Everywhere	Moreover	Since	What
An	Except	Most	So	Whatever
And	Few	Mostly	Some	When
Another	First	Much	Somehow	Whence
Any	For	Must	Someone	Whenever
Anyhow	Former	My	Something	Where
Anyone	Formerly	Myself	Sometime	Whereafter
Anything	From	Namely	Sometimes	Whereas
Anywhere	Further	Neither	Somewhere	Whereby
Are	Had	Never	Still	Wherein
Around	Has	Nevertheless	Such	Whereupon
As	Have	Next	Than	Wherever
At	He	No	That	Whether
Be	Hence	Nobody	The	Whither
Became	Her	None	Their	Which
Because	Here	Noone	Them	While
Become	Hereafter	Nor	Themselves	Who
Becomes	Hereby	Not	Then	Whoever
Becoming	Herein	Nothing	Thence	Whole
Been	Hereupon	Now	There	Whom
Before	Hers	Nowhere	Thereafter	Whose
Beforehand	Herself	Of	Thereby	Why
Behind	Him	Off	Therefore	Will
Being	Himself	Often	Therein	With
Below	His	On	Thereupon	Within
Beside	How	Once	These	Without
Besides	However	One	They	Would
Between	I	Only	This	Yet
Beyond	Ie	Onto	Those	You
Both	If	Or	Though	Your
But	In	Other	Through	Yours
By	Inc	Others	Throughout	Yourself
Can	Indeed	Otherwise	Thru	Yourselves

Sample Suffix List.

-abilities
-ability
-able
-abled
-abledly
-ableness
-ablenesses
-abler
-ables
-abling
-ablingful
-ablingly
-ably
-aceous
-aceously
-aceousness
-aceousnesses
-acies
-acidous
-acidously
-aciousness
-aciousnesses
-acities
-acity
-acy
-ae
-age
-aged
-agedly
-ager
-ages
-aging
-agingful
-agingly
-aic
-aical
-aically
-aicals
-aicism
-aicisms
-aics
-al
-alisation
-alisational
-alisationally
-alisations
-alise
-alised
-alisedly
-aliser
-alises
-alising
-alisingful
-alisingly
-alisings
-alism
-alisms
-alist
-alistic
-alistically
-alisticism
-alisticisms
-alistics
-alists
-alities
-ality
-alization
-alizational
-alizationally
-alizations
-alize
-alized
-alizedly
-alizer
-alizes
-alizing
-alizingful
-alizingly
-alizings
-alled
-alledly
-allic
-allically
-allicism
-allicisms
-allics
-alling
-allingful
-allingly
-allment
-ally
-alment
-alness
-alnesses
-als
-ance
-anced
-ancedly
-ancer
-ances
-ancial
-ancials
-ancies
-ancing
-ancingful
-ancingly
-ancings
-ancy
-aneous
-aneously
-aneousness
-ant
-antaneous
-antaneously
-anted
-antedly
-antialities
-antiality
-antialness
-antialnesses
-antic
-anticism
-anticisms
-antics
-anting
-antingful
-antingly
-antings
-antly
-antment
-antments
-antress
-antresses
-antry
-ants
-ar
-arial
-arials
-arian
-arians
-aric
-aricism
-aricisms
-arics
-aries
-ariliness
-arily
-ariness
-arinesses
-arisabilities
-arisability
-arisable
-arisation
-arisations
-arise
-arised
-arisedly
-ariser
-arises
-arising
-arisingful
-arisingly
-arisings
-arism
-arisms
-arist
-aristic
-aristicism
-aristicisms
-aristics
-arists
-arities
-arity
-arizabilities
-arizability
-arizable
-arization
-arizations
-arize
-arised
-arisedly
-arizer
-arizes
-arizing
-arizingful
-arizingly
-arizings
-arly
-aroid
-aroids
-ars
-ary
-asis
-asise
-asiseable
-asised
-asisedly
-asiser
-asises
-asising
-asisingful
-asisingly
-asisings
-asizable
-asize
-asized
-asizedly
-asizer
-asizes
-asizing
-asizingful
-asizingly
-asizings
-asm
-asms
-ast
-astic
-astical
-astically
-asticism
-asticisms
-astics
-astment
-astments
-astries
-astry
-asts
-asy
-ata
-atabilities
-atability
-atable
-atables
-atably
-atal
-ate
-ated
-atedly
-ately
-ateness
-atenesses
-ater
-ates
-atic
-atical
-atically
-aticism
-aticisms
-atics
-ating