The Excalibur TREC-4 System, Preparations, and Results

Paul E. Nelson, VP of Text Products
(410)-740-8800
paul_nelson@cq.com

Background

This paper describes the system, preparations, and results for the Excalibur text retrieval system used for the TREC conference. After a brief company history and background, we will discuss the system architecture, TREC-4 preparation, an analysis of our results, and some general conclusions.

This will be the third Text REtrieval Conference (TREC) attended by the Excalibur development team. For TREC-1 and TREC-2 we participated as part of ConQuest Software. Please refer to the earlier conference proceedings (available from the National Institute of Standards and Technology) for a discussion of our earlier results in these first two conferences. We did not participate in TREC-3.

ConQuest merged with Excalibur Technologies Corporation (the combined company is Excalibur) in July, 1995, just before the TREC-4 results were due. We are fortunate to have some additional resources to devote to query evaluation, to accuracy studies, and to our preparation and participation in the TREC conferences.

Officially, the ConQuest server system is now part of a larger product suite available from Excalibur called RetrievalWare. We ran TREC-4 with early versions of RetrievalWare 5.0, which is now fully released and available as of December 1995. Many of the improvements and advances which we discovered as part of our TREC-4 evaluations (term grouping, new semantic network weights, a new relevancy ranking formula, and new fine-rank weighting windows) have now been incorporated into Version 5.0 of the product.

ConQuest Software was founded in 1990 with the goal of using Natural Language Processing (NLP) and available linguistic data (such as dictionaries, thesauri, and semantic networks) to produce text retrieval products with high accuracy and performance for large databases and large user populations.

Excalibur was founded in 1980 to create products that utilize Adaptive Pattern Recognition Processing (APRP™) to resolve user queries even in the face of unpredictable and erroneous data (in particular, errors due to the process of Optical Character Recognition when scanning and loading paper documents).

The scope of both companies has grown over the years. The combined company, Excalibur, now has products for document management and high-performance text retrieval, for workgroups, enterprises, and on-line systems. Adaptive Pattern Recognition Processing, in addition to being fully incorporated into RetrievalWare, is also being applied to similar TREC-like tasks on images, such as fingerprint recognition, face recognition, and positive ID.

Finally, we would like to give many fervent and heartfelt thanks to the RetrievalWare development team in the Excalibur Columbia office, who committed long hours and seemingly inexhaustible energies to the TREC-4 task.

Retrieval System Architecture

This section gives a brief description of the RetrievalWare system architecture used for TREC-4. Additional detail can be found in our earlier papers for TREC-1 and TREC-2, or by calling Excalibur Sales at (703)-790-2110.

The basic components of RetrievalWare are shown below:

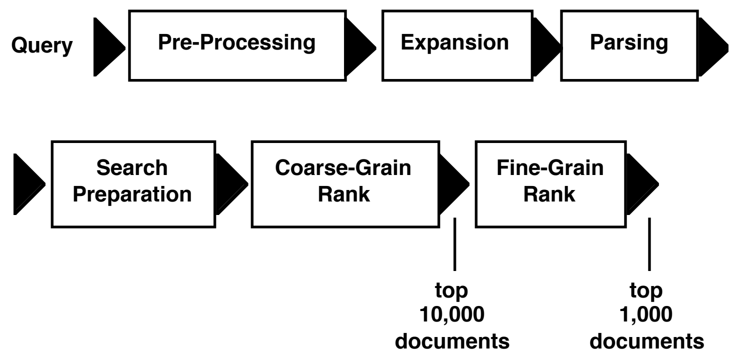

Figure 1 The RetrievalWare Search Engine

Each of the blocks has a well-defined purpose and contribution to search accuracy or performance:

Pre-Processing

Perform query string pre-processing steps to enhance and clarify the characters, tokens, and words in the query. This includes:

Tokenization - Dividing strings into words.

Dictionary Lookup - Locating words in the dictionary and augmenting the words with dictionary information (such as meanings, part-of-speech, inflections, other words with common roots, and variant spellings).

Word Reduction - When a word is not found in the dictionary, rules are applied to reduce the word to a root or simpler form which can then be found in the dictionary.

Token Typing - Identifies the types of special tokens in the query, such as numbers, expansion operators, or other query operators.

Expansion

Apply one of the following expansion techniques to the word. Version 5.0 only applies one of these expansion techniques to any given word (this can be controlled by the user). Future versions will likely apply multiple techniques. Note that the query structures produced by these techniques may not necessarily be the same.

Semantic Expansion - Meanings of the word are expanded to include additional words which are semantically related to the meaning.

Fuzzy Spelling (APRP™) - The word is expanded to include additional words which are similarly spelled.

Wildcards - The user can specify word patterns using wildcards. Words which match the pattern are added to the query, up to a user-defined limit.

Parsing

Query parsing is required to handle a wide variety of special query operations supported by RetrievalWare. These include the following:

Numeric and Date Ranges - Allows the user to search for numeric or date ranges in the body of the document, including open-ended ranges such as greater-than and less-than.

Term Grouping / Query Structure - The user can create statistical groups of terms which are combined together before being combined with other groups or terms. This grouping provides query structure which enhances the “completeness” and “contextual evidence” factors of the query. See the discussion on relevancy ranking below for more details.

Boolean Expressions - Standard boolean queries are supported, including AND, OR, NOT (unary and binary), parenthetical groupings, WITHIN (word proximity in any order), and ADJ (word proximity with order enforced).

Fielded Queries - Any query can be applied to any pre-defined field or zone of a document. Fields or zones of documents are parsed during document indexing with SGML, by using the RetrievalWare document parser, or as defined by a document loading program.

Exact Phrases - Searches for a simple sequence of terms, in the proper order, with all terms adjacent to each other.

User Feedback

At this point the user can view how the query was constructed and expanded and can modify it to better represent the objective. At this time, tools are available to do the following: 1) delete words, 2) choose meanings of words, 3) choose weights for terms, and 4) delete word expansions. The TREC-4 experiments utilized all of these user tools.

Note that this feedback is performed **without** referring to the text of any document retrieved (this is required by the rules for the Adhoc test in TREC). Instead, the query is pre-processed by RetrievalWare and these adjustments are made before the query is executed against the database.

Search Preparation

The search preparation phase is essentially the setup for executing the query. This includes allocating the necessary memory, looking up all words in the indexes, and checking to make sure that all words are found.

Coarse-Grain Rank

Many text retrieval systems use two passes to retrieve documents and RetrievalWare is no different. Our first pass is called Coarse-Grain rank, and it is used to reduce the search set from the entire search database down to a manageable number of documents (typically a few thousand for the highest accuracy, or a few hundred for higher performance). Once the search set is thus reduced, each document can be processed with more accurate statistics with Fine-Grain rank. The two-pass technique is used solely to improve search speed.

The Coarse-Grain ranking algorithm is necessarily more inaccurate than fine-grain ranking, primarily because it considers only the presence or absence of a term in a document (without considering the proximity of words to each other, or the frequency of terms within a document). Coarse-Grain ranking includes the following functions and statistics:

Expansion Term Distance - Documents with terms that are more closely related (by spelling distance or semantic distance) to the original terms in the query are weighted higher.

Query Structure, Completeness, and Contextual Evidence - Once the set of terms in a document is known, the weights of the individual terms are combined using the structure in the query (sets of terms and the functions applied to these sets).

Our algorithms automatically produce structure which enhances **completeness** (documents which contain representatives from all main terms in the original query are weighted higher) and **contextual evidence** (terms are weighted higher when they are supported by expansion terms that are semantically related to meanings chosen by the user).

Main Term Weighting - The term weights chosen by the user, or through automatic statistical techniques, are used to combine term weights together, for the terms which occur in a document.

Fine-Grain Rank

Once a subset of documents has been found by Coarse-Grain rank, Fine-Grain rank uses all additional data to come up with the final set of document statistics. The additional data is primarily the list of positions for every query word in every document. This data is used to compute several statistics, including the strength of each term (based on how far away the term is from other query terms), the average strength of all terms for each document, the maximum term strength for a document, the number of query term occurrences in the document, and a simple count of all the terms in the document.

Finding the strength of each term in the document is (frankly) an area of active research. We experimented with a large number of functions for TREC-4. All functions re-use the same query structure and term weights used for coarse-grain ranking. However, the terms may be combined with different functions, and the term weights may be attenuated based on a proximity weighting window (terms near the edges of the window typically have lower strengths).

Many architectural decisions on RetrievalWare were based on factors which had nothing to do with text search accuracy. These include, royalty requirements and product rights to available dictionary and semantic network information, speed of indexing, requirements for simultaneous indexing and query, and index compression factors.

The most important architectural decision was to make RetrievalWare a full parallel processing text search system that can distribute queries across multiple physical machines and databases in a Local Area Network or Wide Area Network. Distributed queries requires that certain global database statistics be avoided, so that documents retrieved from multiple parallel engines can be easily and properly merged into a single result list for the user. In particular, our system eschews the Inverse Document Frequency statistic (the total number of documents which contain a word, divided by the total number of documents in the database).

Preparation for TREC-4

We strove for two basic goals, which we felt would improve our accuracy the most in preparation for TREC-4: 1) run as many tests as possible to test many variations, and 2) constantly question and refine the results analysis. Thus our approach was more methodical and experimental, rather than theoretical. The raw performance of our engine allowed us to run hundreds of tests and tens of thousands of queries. This allowed us to use some clever numerical techniques to fully optimize search parameters.

All accuracy tests (except for the final results submission, of course) were performed with data from TREC-3 data exclusively. Queries were created using only the "Description" field from TREC-3, to most closely simulate the queries from TREC-4 in size and content. We were worried that we might be hurt by tuning to TREC-3 queries since they are slightly longer than TREC-4 queries (10-15 words instead of 5-10 words), but it appears that this had minimal affect. (See the next major section describing our final test runs for more details on how the final queries were generated.)

All through our preparations, we wanted to double-check our results by testing against TREC-2 queries to make sure that our modifications were not being biased by the TREC-3 query suite. Unfortunately, we were only able to run this double-check once, near the end of our preparations, and the results were (at the time) inconclusive.

The following subsections describe the kinds of experiments that we ran.

Relevancy Ranking

Much experimentation and statistical analysis was directed towards finding the optimal combination of the basic output statistics from the query engine: Maximum Hit, Coarse Rank, Old Fine Rank, and Hit Percentage (Number of Hits / Document Length).

Many techniques were tried for determining "optimal" coefficients, including: 1) a statistical regression over all queries (similar to probabilistic query techniques), 2) regressions with "normalized" statistics (attempting to account for differences between queries), 3) regressions over each query followed by averaging the coefficients, and 4) numerical/iterative techniques to search for the optimal combination. The best technique used the averaged coefficients, with slight adjustments made by iterating the coefficients.

The final function was roughly 4.0*(hit percentage) + 0.59*max_hit + 0.76*coarse_rank. Unfortunately, that regression formula often provides numbers greater than 1. This is because we used linear regressions to compute the coefficients, which attempted to predict a 0 or 1 event (the document relevancy). The regression necessarily overcompensates for the 1.0 case, and so minimum error occurs when the best documents are ranked greater than 1.0.

The five statistics mentioned at the top of this subsection were not the only numbers with which we experimented. We also tried many other functions, including density functions, density overlaps, judging the importance of each term to the document, inverse document frequency, and simply taking the log of the existing statistics. With all these different numbers, there was considerable debate about which ones to use. We tried several techniques for determining which input variables would produce the best results, (such as using the R-squared values to guide our selections, etc.), but no algorithm appeared to be very reliable. Remember that the computation of the coefficients is more complicated than normal because the 50 individual queries were regressed separately and then the coefficients were averaged together, which (we believe) prevented standard techniques from helping much. Ultimately, we chose the statistics above based more on a gut feeling rather than any repeatable technique, but even so we had gained a fair confidence that these were good ones. Obviously, more research is required.

Controlling Expansions

One way of significantly improving our accuracy was to more carefully control which semantically-related expansion terms were added to the query. This appeared to improve our scores by about 4 percent or so (a strong contributor).

Three techniques were used to control the expansions of semantically-related query terms. First, users (query writers) were asked to choose the most appropriate meanings for each significant (i.e. non-stop) word in the query. This limited expansions to only terms which were semantically related to the chosen meanings, based on how the original query terms were used in the query.

The second technique allowed the user to further choose from the list of actual expansion terms for each query term. As it turns out, the expansions for a term contain many subtle shades of meaning, and choosing from these shades of meaning can considerably improve the search accuracy. For example, “murder” expands to “assassinate”, which is probably not useful unless the query has a political connotation. A second is example is that “kill” expands to “suicide”. This technique is especially useful at high expansion levels.

The third technique was to change the weights on the links between the words. Changing the weights not only changes the strengths of the expansion terms, but if the strength is too low (lower than the expansion threshold) then the term will be removed from the query completely. We tried several methods for optimizing the weights. One technique created a TREC-like database of related terms used to evaluate the relevancy of our expansion terms. This database was used to evaluate expansions from all link types, and the weights were set based on the number of relevant terms retrieved. The terms used for this test were based on TREC-2 query terms. This is the technique we used for our formal TREC-4 tests.

Other techniques for evaluating expansion terms included iterating the weight on each link type and then re-running the entire test (which took many days to complete). We also tried simple global modifications such as multiplying all weights by 2, 3, and dividing them. None of these other techniques produced a significant improvement.

Term Weighting

Term weights for our TREC-4 system came from two sources: First, weights for expansion terms were determined by the links traversed in the semantic network (see above). Second, all terms could be manually weighted by the query writer. Careful term weighting appeared to improve the scores significantly, about 5-10 percent.

Initially, we were very skeptical of manually weighting terms, since evaluation of our TREC-2 results showed that manual term weighting did not improve (and may have hurt) the overall system performance. In particular, statistical analysis showed that documents which contained terms that were weighted higher by the query writers had little or no additional probability of being relevant. Needless to say, this result was non-intuitive, and it now appears to have been wrong, since the results from TREC-4 are unmistakable. Most likely, our TREC-2 queries were just poorly weighted, or perhaps our algorithms for evaluating probability of relevance had bugs in it.

Fine-Grain Ranking

Fine grain ranking actually considers the positions of the query terms within the documents retrieved by the coarse-grain ranking function. Terms which are in close proximity are ranked higher by the fine-grain function.

The fine-grain ranking algorithm calls for a window to be passed over the document (only the indexes are required for this step, the document itself is not actually retrieved). For every window position, the query words which occur within the window are combined together (using the same functions as for coarse-grain rank) and the strength of the position is determined.

To further enhance the proximity test, the terms near the center of the window have a higher strength than terms near the edges of the window. A sample window attenuation function is shown below.

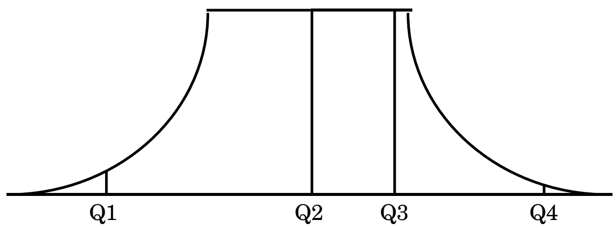

Figure 2 Sample Window Attenuation Function over Four Query Words

In preparing for TREC-4, we experimented with many fine-grain ranking functions, mostly having to do with wider weighting windows and less attenuation of terms which are away from the center of the window (the plateau shown above). Most experiments showed little or no improvement. We ended up using a window which showed a small 2-3% improvement in the results.

The existing system is limited to windows of approximately 200 words wide. Later versions of RetrievalWare may allow us to expand the window to greater than 200 terms (without loss of search accuracy), which may yield further improvements.

Query Structures

Many tests were devoted to improving the structure of queries. In RetrievalWare, queries are not simple lists of terms (as in some Vector approaches), rather queries have structure. RetrievalWare 5.0 was the first version of the software to allow an arbitrary hierarchical structure to the queries (previous versions limited statistical queries to only two levels of hierarchy). Further, RetrievalWare 5.0 allows the integrator to choose the operators (boolean or statistical) which occur at any hierarchical node.

The query structure which helped the most was term grouping, which combines multiple related terms into sets. Each set is ranked individually (as if the set was retrieved from the dictionary). This technique seemed to improve the queries by 5-8%, and also gave us the flexibility to add terms to the query which we might not have added before.

Term grouping is a query structure which essentially enhances the contribution of completeness to the accuracy algorithms. When terms are in groups, the query engine emphasizes documents which have at least one term from each group. Additional terms from the same group do not add as much strength as would additional terms from different groups.

Term groups in RetrievalWare were created manually, and only for the CnQst2 run. Of course, the dictionary naturally creates term groups made up of related terms. This additional grouping is for terms which would not have been retrieved from the dictionary (or for related terms in the query itself).

Things Which Didn't Work

For every test which worked, there were (easily) 30 tests which did not work. More often than not, we would end up with inconclusive data, meaning that the results did not improve or degrade by a significant amount.

Among other things, we were unable to show positive benefits with any of the following:

- Inverse Document Frequency (IDF)
- Expansion normalization
- Other coarse-rank functions (max, min, stronger, weaker, etc.)
- Dozens of other results sorting algorithms
- Miscellaneous hit density functions

The most surprising result here is that IDF showed no reliable improvements to the results. We tried many different ways of weighting terms with IDF, and none seemed to help much. It was suggested at the conference that our other techniques (such as controlling expansion terms, term weighting techniques, and term grouping) may have reduced the need for IDF. It is likely that this is true.

Our Final Test Runs

Excalibur participated in the Adhoc, manually generated, query test over the Category A data. For readers who are unfamiliar with the TREC tests, the "adhoc" test specifies that queries are generated without the benefit of reading documents (i.e. no document feedback), the queries are executed once, and then results are sent to the TREC committee. "Manually generated" simply means that query creation had some human input. "Category A" data is the entire TREC-4 □data set, roughly 2.5 GB of text data.

Excalibur submitted two query runs which differed in the amount of manual intervention in query construction. These two query runs are described in the next two subsections.

Query Construction Test 1 (Run ID CnQst1)

The first query construction test (labeled CnQst1) was similar to RetrievalWare expert mode queries: limited to choosing meanings, weights, and expansions. The original TREC-4 queries, with no modifications, were entered into the search system. After pre-processing, the user was given the opportunity to choose the relevant meanings of each word. Then, after expansion, the user was given the opportunity to remove any irrelevant expansion terms from the query. Finally, the user was allowed to change the relative weightings of the terms in the query.

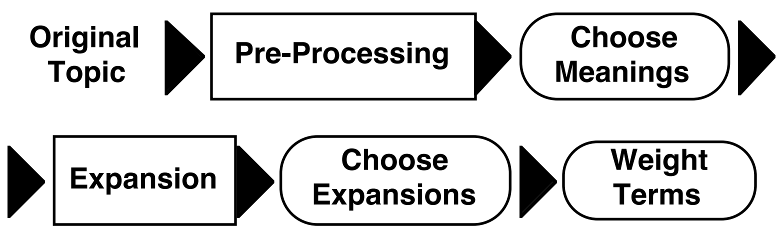

Figure 3 Query Construction for the CnQst1 Run

This method of query construction was chosen because it most closely resembles the standard capabilities provided with RetrievalWare 5.0, as if the user entered exactly the query specified by the TREC committee. Graphical user interfaces are provided for each of these steps.

In our experience, choosing meanings and choosing expansions both turn out to be pretty obvious and straightforward activities. For most words and meanings it is usually clear what is relevant or not.

Term weighting, however, is more complicated and required some experimentation. It was important to identify the terms which were essential to the query and those which were merely supportive. This drove the term weights which were assigned.

Each query took about 5 minutes to create.

Query Construction Test 2 (Run ID CnQst2)

The second run submitted by Excalibur allowed additional flexibility in constructing the queries. Specifically, the query writer was allowed to add terms to the query and was allowed to group terms into phrases or statistical sub-groups (see the discussion of query structures above).

Additional terms were added to the query at the discretion of the query writer. These terms were mostly obvious omissions from the original query, although some terms provided additional examples or supporting information. The query writers did **not** have access to any additional resources when generating these additional terms. In particular, they did **not** use any reference materials or documents. Only terms which occurred to the developers while writing the queries were added.

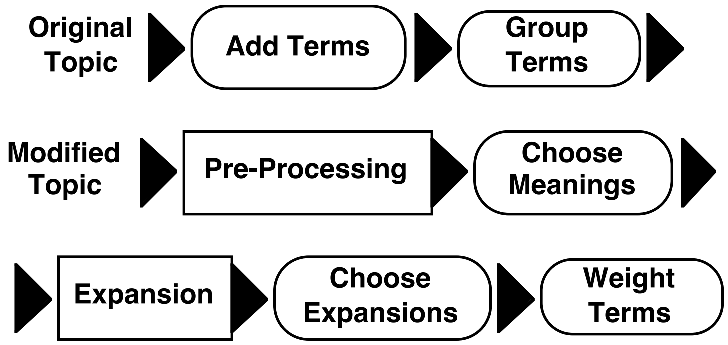

Figure 4 Query Construction for the CnQst2 Run

Term grouping allowed query writers to add more terms than normal to the query, as long as the additional terms were grouped with other terms into main conceptual groups. Basically, a term group behaves the same as if the terms were retrieved from the dictionary as expansions on a concept. The same relevancy ranking formula (essentially a probabilistic combination) is applied to both situations. Since the formula is linear, this means that dictionary expansions of terms within a set are treated simply as if all the expansion terms were part of the set.

Grouping query terms was pretty easy, since the query writers simply created a group for each main concept in the query. Additional query terms were then added to the most appropriate group. Since the TREC-4 queries were so short, identifying the main concepts was quick (the same technique in TREC-2 was much harder, for example, since the topics were so much larger and more complex).

Each query took about 8 minutes to create.

The following are two sample queries used for TREC-4:

```
(recycle:3 recyclable recyclablity retread convert) tires:3 (economic:2
economical:2 economy:2 profitable:2 cost) (Bandag landfill impact)

("bio conversion":5 convert:2 conversion:2 transform:2 generate generation
produce production cogener* "co generation")
```

Conclusions

The following two charts show the performance of both our runs, CnQst1 and CnQst2, in relation to all Manual, Adhoc, Category A systems:

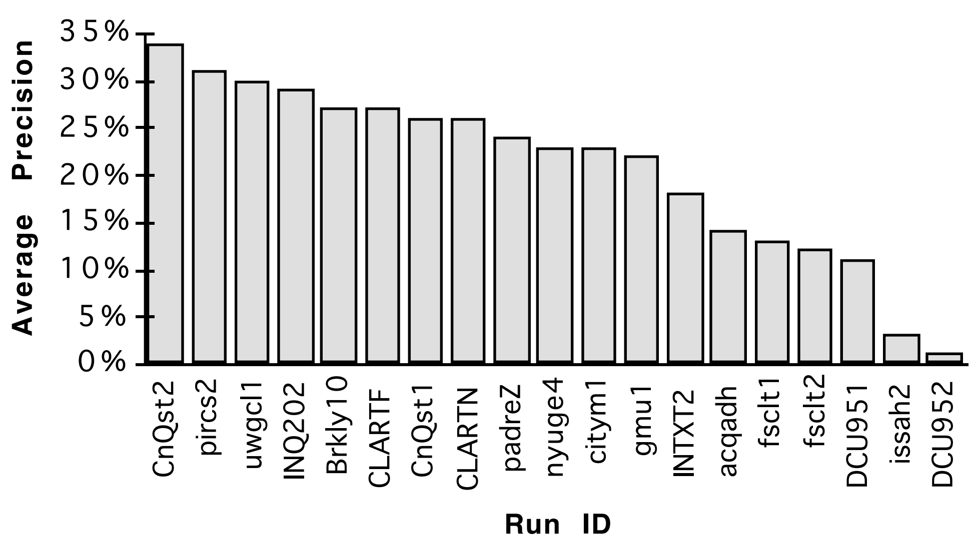

Figure 5 Comparison of Average Precision for all Manual, Adhoc, Category A Systems

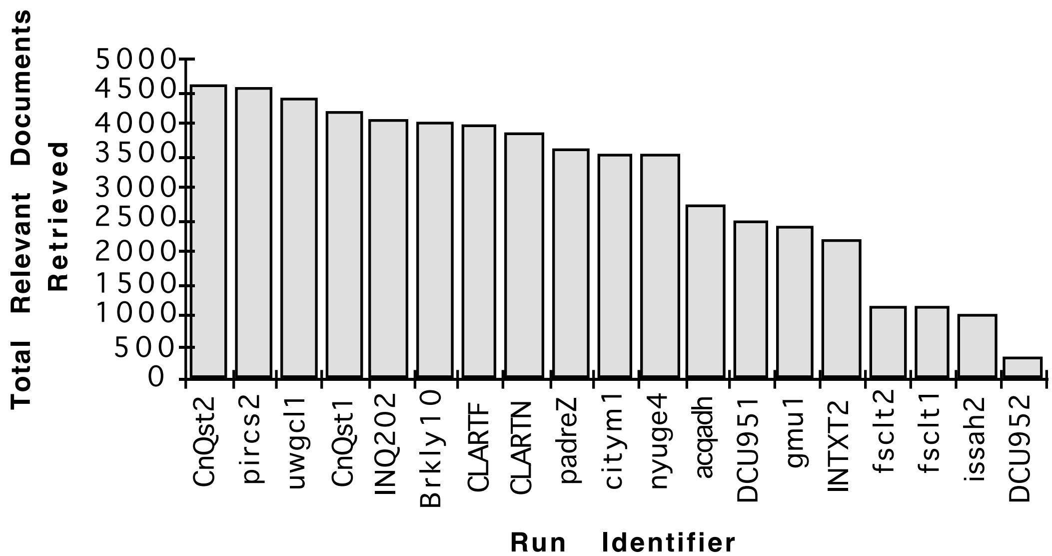

Figure 6 Comparison of Total Relevant Documents Retrieved (Overall Recall) for all Manual, Adhoc, Category A Systems

In TREC-2, Excalibur/ConQuest performed better in Recall (the chart of Total Relevant Documents Retrieved above) than in average precision. Therefore, we were surprised to see that the opposite was true in TREC-4. It appears that our precision has improved considerably, which indicates that our in-depth work on relevancy ranking formulae was worthwhile.

In looking at the two graphs above, it is interesting to note that the CnQst1 run was better in total documents retrieved (recall) than in average precision, in relation to other systems. In fact, it was the only run in the top 9 which changed position.

This leads to another comparison. The difference between CnQst1 and CnQst2 in overall recall appears to be just 9.3%, whereas the total difference between the two runs in average precision was 23.5%. This is pretty surprising. We naturally expected that the CnQst2 queries would be much better in recall, due to the added terms. This implies that additional query terms did not help that much (9% improvement in recall), but that the query structure (i.e. the grouping of terms) helped a lot (24% improvement in average precision). Of course, the difference could simply be due to the scale of the measurements or the sensitivity of average precision (see the next paragraph). This idea will need to be tested further.

During our preparations for TREC-4, we used the total number of relevant documents retrieved as the determining statistic. This is shown above in Figure 6 (there were a maximum of 6501 relevant documents available to be retrieved from the database). We felt that the average precision score was too sensitive to unread documents, especially at the top of the returned results list. This observation came from a comparison of our system variations, where the total relevant documents returned would generally increase while the average precision would vary widely. However, it is unlikely that this instability is responsible for the difference between total recall and average precision in our final runs, since the top 100 documents of each run were read and evaluated by the TREC assessors (and hence the problem of unread documents at the top of the range is minimized).

Two additional comments:

- Only 1 out of every 30 tests that we ran in preparation for TREC-4 actually improved accuracy. Most tests showed little or no performance improvements.
- It appears that the Excalibur/ConQuest relevancy ranking has always performed better on shorter queries. It is certainly the case that our early designs emphasized this fact, primarily because the major on-line customers claimed that the average query was a two or three term phrase. To some extent, this explains our better performance in TREC-4 where the queries were smaller, as opposed to TREC-2.

Some ideas for future tests include the following: 1) automatic term expansion, 2) automatic term weighting (using TREC-3 and TREC-4 queries as the "ideal" for generating proper weights), 3) more sophisticated query structures derived from the semantic networks, 4) additional testing and statistical analysis. In particular, we expect to start looking at the behavior of the 50 TREC-3 queries in more detail, to see if certain types of queries behave in statistically similar ways. Our feeling is that many of our evaluations fail to show improvement because some queries get better while others get worse, and so the average performance is washed out. If we can categorize the queries into sets which perform in similar ways, this may lead to cleaner evaluations of our ranking functions and statistics.