The FDF Query Generation Workbench

K-I. Yu **P. Scheibe** **F. Nordby**

Paracel, Inc.
Pasadena, CA

The TREC-2 experiments with the FDF-3 clearly showed the power of automated query generation from relevance information; the broader TREC-2 results also suggest that there may be no single "best" query generation method. Building on this experience, Paracel has begun the design and implementation of a Query Generation Workbench for the FDF-3, to provide a structure for implementing and selecting among a variety of relevance-based query generation methods. TREC-3 has provided an early large-scale test of this prototype, and we have used the opportunity to begin a study of the first generation of tools for the Workbench.

1. Introduction

In TREC-2, two methods were used to generate queries for the FDF-3 hardware text search system: for one a human expert manually generated queries for all 50 topics; the other used an automated term weighting scheme, augmented by operator-supplied suggestions. The results of this experiment were clear: the automated system produced superior results in less human time — including the time required to implement the system!

More broadly, the diversity of systems and results at TREC-2 suggests to us that there may not be any one "best" automated query generation system. Certainly there were systems that performed well overall; there were also many systems — including the human-generated FDF-3 queries — that showed superior performance for some specific topics.

Based on these observations, Paracel has begun the design and implementation of a Query Generation Workbench. This Workbench has two complementary aspects: in the broad sense, the Workbench is a structure for the implementation, evaluation and combination of different query generation methods; in the narrow sense, the Workbench comprises a collection of tools which can be combined within this structure to implement different query generation variants, and a query selector which can be used to automatically choose among a variety of query generation methods based on estimated performance for a particular query generation task.

TREC-3 has provided an early large-scale test of the Workbench. For this first test we populated the Workbench with a variety of conservatively-designed and -implemented tools; then we set the system on automatic and let it run its course. Thanks to the modular structure of the Workbench, and thanks to the operation of the query selection stage of the Workbench, we immediately derived an extra result: in addition to a set of queries, the selector returned a parameter-by-parameter, tool-by-tool, topic-by-topic comparison of the different query generation tools in the Workbench. We have begun an analysis of these data, and the preliminary results have shown some interesting patterns. We are now working to incorporate these results into the Workbench and its component tools.

```
{ 1 document -> return 1+ (
    'guilty' : 12 ;
    'insider''trading' : 21;
    'ivan''boesky' : 26 ;
    'trading' : 18 ) }
```

Figure 1. A simple FDF weighted set query. If the term "`'guilty'`" is matched once or more in a document, then 12 is added to the score for that document; similarly, if "`'ivan''boesky'`" is matched, 26 is added to the document score. With this query, the maximum score for any single document is 77 (i.e., 12+21+26+18).

2. The Query Generation Workbench

The goal of Paracel's FDF Query Generation Workbench project is to develop an "industrial strength" system for automating query construction from relevance data. While this system is oriented towards fully automatic query generation from relevance data, it also allows for user input at various stages of the query generation process. Finally, since the TREC results suggest that there may be no one "best" relevance-based query generation method, the Workbench makes provision for case-by-case selection among a collection of query generation methods and parameters.

The major early result of this project is a prototype Workbench architecture. This architecture is a uniform, modular framework for relevance-based automatic query generation. The Workbench structures query generation in stages, and provides plug-and-play use of different tools at each stage, allowing piece-by-piece customization of the query generation process. Also, since one of the stages in the Workbench is a query selection, or simple query-level data fusion, the Workbench allows for automatic parameter optimization and tool selection within the same framework.

The Workbench structures query generation as a four-stage process. This process starts with the enumeration of candidate terms: these terms may be anything for which the FDF search engine can match. From the candidate terms, a subset is selected for use in the query; then the selected terms are weighted and built into an a query for the FDF search engine. Finally, given a collection of queries — typically the collected results from various tools and parameter values in the first three stages — the performance of each query is estimated, and one query is selected for actual use as routing profile. In summary, the query formulation stages used in the Workbench are:

1. Enumerate a set of terms (words, word pairs, etc.).
2. Select a subset of the terms enumerated in step 1 for use in the final query.
3. Choose a weight for each term selected in step 2 and use these weights to formulate a query.
4. Given several queries using a variety of selection and weighting criteria in steps 1, 2 and 3, estimate the accuracy of each of these queries and choose the best.

In the first version of the Workbench, the stages have been populated with tools based primarily on a probabilistic model, and the tools currently all produce queries in a single form most suited to this model (figure 1 for a sample query). We are now working to incorporate methods from vector-space and other models into tools, so that we may use these methods within the framework of the Workbench.

2.1 Candidate Term Enumeration

The first stage of the Workbench is term enumeration: this stage simply lists a collection of relevance cues for possible use in query formulation. The terms produced by the present term enumeration tool are quite simple: words and word sequences[1] from the text of known relevant documents. In this process, case distinctions are ignored, but no stemming is performed. The enumeration tool is parameterized by the lengths of the allowed word sequences.

2.2 Term Selection

Given a list of candidate terms, the next stage of the Workbench is term selection. The first step in this process is stop word elimination: to this end, any term which either begins or ends with a "stop word" is discarded. For the TREC experiments, a "stop word" is either any word from a list of common English words, or any word found in more than half of the documents of any one of the TREC corpora.

Next, the term selectors gather data on the occurrences of the candidate terms, both in the relevant set and in the training database as a whole. These collected statistics are used to calculate a "selection score" for each of the candidate terms by any of a number of formulae, and the terms are sorted according to score. The prototype toolkit includes several term selection functions; for the TREC experiments we used two of these:

1. The odds of document relevance given term presence ("O"): $P(R \mid w_i = 1)$;
2. A heuristic ("VxLD") based on the normalized deviation of the term occurrence frequency and the in-document frequency.

Finally, given this ranking of the terms, some number are chosen from those with the highest scores: this selection process may impose other restrictions, such as a limit on the number and lengths of phrases used in a query.

2.3 Query Construction

The third stage of the Workbench is query construction. All of the query construction tools which currently populate the Workbench produce queries of the same general form (e.g. figure 1). To build queries, the tools use many of the same statistics gathered in the term selection process: from these data, the tools weight each selected term, then build a query is built to approximate of the desired scoring function. For the initial tool set we implemented a few simple term weighting functions; for TREC-3 we used two:

1. The odds of relevance given term presence ("O"), $P(R \mid w_i = 1)$; and
2. The log of the relative likelihood of term presence vs. document relevance ("LLR"), $\log\left(\frac{P(w_i = 1 \mid R)}{P(w_i = 1 \mid \bar{R})} \cdot \frac{P(w_i = 0 \mid \bar{R})}{P(w_i = 0 \mid R)}\right)$.

1. A.K.A. "phrases". This terminology should not be taken to suggest that these sequences necessarily have any linguistic significance; these "phrases" are just sequences of words which appear in the source text.

	Query #1	Query #2	Query #3	Query #4	Query #5
Subset #1	19 (1.0)	16 (3.0)	15 (5.0)	16 (3.0)	16 (3.0)
Subset #2	12 (5.0)	14 (2.0)	13 (4.0)	14 (2.0)	14 (2.0)
Subset #3	17 (5.0)	19 (3.0)	18 (4.0)	20 (1.5)	20 (1.5)
Subset #4	22 (1.0)	20 (3.5)	20 (3.5)	21 (2.0)	18 (5.0)
(Total)	**(12.0)**	**(11.5)**	**(16.5)**	**(8.5)**	**(11.5)**

Table 1. Query selection table for five methods on topic #125. Each column of the table shows the results for one query formulation; each row shows the results for one subset of the corpus. The entries (e.g., "14 (2.0)" for query #2, subset #2) show the recall at 50 documents retrieved for that query on that subset of the training set, and the ordinal rank of that result over all five query formulations. The final row, labeled "(Total)," shows the summed ranks for each query. The query selector picks the query with the lowest summed rank — in this case, query #4.

There are also a number of adjustments which may be made to the term list and the term scores in this stage. The first, and simpler, of these adjustments is a heuristic which we implemented to test the effect of adding a bonus for multiple term occurrences. To this end we doubled size of the query: under this adjustment if a term is found once in a document, the standard term weight is added to the document score; if the term is found twice or more in a document, a 50% bonus is added.

The other term weighting adjustment which is available is a compensation for the presence of sub-phrases in a query. For example, consider the query shown in figure 1: if a document includes a match for "`'insider''trading'`" then it certainly includes a match for "`'trading'`" — thus the presence of the sub-phrase adds no new information. If this query were subjected to phrase weight adjustment, the weight of the superphrase ("`'insider''trading'`") would be decreased by the combined weights of all its sub-phrases included in the query — the final weight would be 3 in this case.

2.4 Query Selection

The final stage in the Workbench is query selection: this stage might described as simple query-level data fusion. The inputs to this stage are any number of queries for the same topic, built with different term enumeration, term selection and query construction methods: from these queries, the Workbench selection tool chooses one query which shows the best estimated performance on the training collection.

To accomplish this, the selection tool divides the test collection into four subsets. Each of the candidate queries is run against the four training subsets, and the performance of each query on each subset is measured as the recall at 50 documents retrieved. The queries are assigned ordinal ranks for each subset, and these ordinal ranks for each query are summed to form an overall query score. The query with the lowest score is then selected. Table 1 shows an actual selection table for one of the TREC-3 topics.

2.5 Practical Matters

The process just described is, in terms of functional results, exactly what is done in the TREC query construction process; however, in practical terms, a number of optimizations are applied. These optimizations impact the performance of the query generation process, but do not reflect on its overall behavior.

For example, a list of all the words in the training corpus is pre-computed and shared among the query construction processes for all topics. Also, word frequency data is collected in the compilation of this word list: this help in determining stop words and computing term selection scores and term weights. This index-style statistics collection works well for single words, but for it would be impractical to maintain statistics for every phrase found in the corpus. Instead, the term selectors compose queries and use the FDF to gather the desired statistics. (The statistics are passed on to the query constructors, so they need not re-collect the data.)

2.6 The TREC-3 Experiments

For this year's TREC experiments we wanted to begin a systematic evaluation of the results from last year's automated FDF query generation results. Accordingly, one of the inputs to the query selector is an emulation of last year's automated query generator: this method phrases up to three words, 50 terms, the "O" selector and the "O" constructor; also, the selector was restricted to only those phrases found in at least 30% of the relevant documents. The method did not use either phrase weight adjustment or a multiple-occurrence bonus.

All of the other inputs to the query selector used the "VxLD" term selector and the "LLR" query constructor with phrase weight adjustment: we had designed these tools to implement a simple log-likelihood model query generator, fixing the "mistakes" in TREC-2 query generator. Here again, the term selector was restricted to those phrases with a recall of at least 30%. In some cases, phrases up to three words were allowed; in others, only single-word terms were included. Term counts were varied from 32 to 64, and in some cases the number of phrases in the final query was restricted. Finally, multiple occur bonuses were used in some of the cases.

3. Results

We have derived two sets of results from the TREC-3 process: first, of course, the formal TREC results provide a gross comparison of the Workbench results to the other TREC systems; second, the data from the query selector show in detail the strengths and weaknesses of the various query generation tools. These latter results in particular are extremely valuable to the tool development process, as they allow point-by-point, parameter-by-parameter, stage-by-stage comparison of different query generation methods.

3.1 Query Selection Results

This year, to provide a reasonably close comparison against last year's results, we included a similar query formulation method as one input to the final query selection step. To this, we added a number of query formulations all based on the VxLD selection and the LLR weighting (with phrase weight correction) formulae; we used these because of limited time, and because our preliminary results suggested that these were over-all the best performers. For these alternative query formulations, we varied several of the secondary parameters: term count, phrase count, etc.

Overall, the VxLD/LLR methods far out-performed the O/O method: seldom was the O/O method chosen by the query selector, and often the O/O method showed the worst performance of all the methods. We have not yet performed an statistical analysis of this comparison, but we expect that the VxLD/LLR methods will be found to be generally superior to the O/O method with strong statistical significance.

That said, there were a few cases in which the O/O query far out-performed the VxLD/LLR queries. We are looking at the terms to try to determine what aspects of the O/O query formulation are responsible for these cases, to see if we can derive a hybrid query formulation method.

Beyond these broad results, there are many more details; to summarize just a few:

- Phrases can be very beneficial, as long as they don't replace good single-word terms in the query.
- Along the same lines, phrases of four words or longer tend to be superfluous: the shorter two- and three-word sub-phrases will already be in the query, and the four-word phrases will add no effective precision to the query.
- The VxLD heuristic for term selection seems to work quite well — the term lists seem, subjectively, appropriate to the topic — and they appear on the whole to provide good recall. More research is needed here, however.
- Similarly, the LLR query builder worked better than the O query builder, on the whole; however, as mentioned above, there were a few cases where the O queries outperformed the LLR queries. We don't yet know wether this is an artifact of term selection or weighting.
- Phrase weight adjustment was a powerful tool — sometimes increasing precision at fixed retrieval levels by a factor of two or more with all other parameters held constant.
- Multiple occurrence bonuses often but not always had a positive effect on query performance. More research and a more rigorous model are needed here. Also, new query construction tools are needed to build queries which reflect a broader range of query generation methods.
- The query selector is too simple: use of precision at 50 documents as ranking criterion should probably be replaced by another (perhaps selectable) metric — for example, one which estimates average precision. Also, some more care needs to be given to ensuring unbiased performance estimation in the query selector.

3.2 TREC Scores

Our analysis of the official TREC results is on-going. In gross terms, the overall performance of the Workbench was approximately median — measured as precision at 1000 documents, for example, the Workbench scored at or above median for 26 of the topics and below median for 24 of the topics. We are currently examining the results in greater detail, and we hope to learn more from that analysis.

4. Conclusions and Acknowledgments

Our efforts have been focussed on developing a framework for query generation, and populating that framework with an initial set of tool. The results have shown us a great deal about the component effects present in the query generation process, and we are learning more as we continue this work.

As we have shown, the avenues for further research and development in the FDF Query Generation Workbench are many and broad. In the main, we are working to incorporate the major lessons learned into the structure of the Workbench and its constituent tools; and we are working to extend the comparisons we began with our TREC-3 experiments. Also, since the FDF search engine is entirely language independent, we are starting to apply the Workbench to non-English query generation problems. Along these lines, we expect to participate in the non-English TREC exercises next year.

We would like to express our gratitude to the TREC sponsors and participants alike. The former have provided an excellent testbed for research and development in text retrieval systems; the latter have sown this fertile ground with ideas. We hope to continue to benefit from this abundance in the future.