Towards Interactive Query Expansion

Donna Harman*

Lister Hill National Center for Biomedical Communications
National Library of Medicine
Bethesda, Maryland, 20209

Abstract

In an era of online retrieval, it is appropriate to offer guidance to users wishing to improve their initial queries. One form of such guidance could be short lists of suggested terms gathered from feedback, nearest neighbors, and term variants of original query terms. To verify this approach, a series of experiments were run using the Cranfield test collection to discover techniques to select terms for these lists that would be effective for further retrieval. The results show that significant improvement can be expected from this approach to query expansion.

1. Introduction

Statistically-based keyword retrieval systems are known for almost always retrieving some documents relevant to a query, but seldom retrieving all documents relevant to that query, at least by a rank most users will see. Many efforts have been made to improve statistically-based keyword performance, but only the use of various types of term weighting for ranking [SPARCK-JONES72, CROFT83, SALTON83, HARMAN86], have made enough improvement to be universally accepted as the state-of-the-art for measuring further improvements. Usually improvements in recall are achieved only at the expense of precision.

One method for retrieving more relevant documents is to expand the query terms by using relevance feedback, conflating word stems, and/or adding synonyms from a thesaurus. These additional terms allow the query to match documents that contain words which are related to the query, but not actually expressed in it. This paper is based on the following premise: a statistically-based keyword system that retrieves a few relevant documents in answer to a query should be able to help the user modify the query in order to retrieve more relevant documents.

As an example, suppose that a user inputs a query, finds several relevant documents in the first screenful of documents shown to them, and then desires to see more documents. The following sample screen might then appear.

*Current address: National Bureau of Standards, Gaithersburg, Maryland, 20899

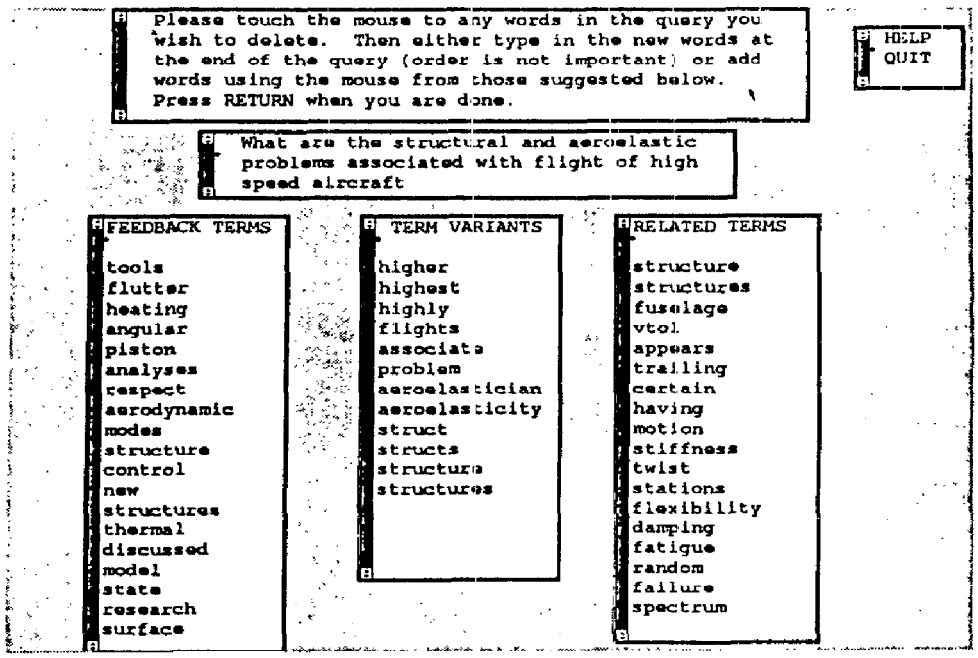

Figure 1 -- Sample Screen for Query Modification

The first window contains words derived from relevance feedback (see below). The second window contains term variants of the original query terms, and the third window contains words derived from a manual thesaurus, or some type of automatic generation of word relations. The size of the windows and the patience of the user require that only a reasonable number of terms be displayed. The goal of this experiment was the discovery of techniques for filling those windows with the terms most likely to increase performance.

2. The use of relevance feedback for query expansion (Window 1)

2.1 Past work

Relevance feedback is an interactive retrieval tool, but can be used in conjunction with a test collection by simulating the user interaction. That is, assuming that the user is shown the first ten documents retrieved, the known relevant documents in that set become the relevant documents used for feedback.

One of the first trials of relevance feedback was by Rocchio in 1965 [ROCCHIO68], where he simulated relevance feedback by using a test collection with known relevant documents. Using the SMART vector approach, Rocchio modified the queries by adding the term vectors for all relevant documents retrieved by a given cutoff, and subtracting the term vectors for all non-relevant documents retrieved by that cutoff. This effectively added many terms, some with negative weights; increased the weights of some query terms; and decreased the weights of others. The results, however, were not conclusive because of problems in evaluating relevance feedback in general.

Wu [WU81] also expanded the queries using the SMART vector expansion method developed by Rocchio, but then reweighted all the query terms using a revised term distribution method developed earlier [YU76]. The most improvement came from reweighting, with a smaller increment from the added terms. Porter [PORTER82] implemented the CUPID retrieval system using a different reweighting scheme [ROBERTSON76] and query expansion, with user selection of the additional query terms from a set derived using relevance feedback, but did no extensive evaluation to the author's knowledge.

Smeaton[SMEATON83] tried three methods of query expansion, one based on nearest neighbors to query terms, one based on maximal spanning trees (a variation on nearest neighbors) for query terms, and one based on terms from documents retrieved using relevance feedback. He found similar performance between maximal spanning trees and nearest neighbor expansion, with worse performance using relevance feedback, and generally little improvement using any method. One problem he noted was the large number of terms added when taking all terms from the retrieved relevant, and he attempted unsuccessfully to select only the "best" terms to add, using several methods.

A modification of Porter and Smeaton's approaches seemed to be the most promising, with the user selecting terms from a small subset of the terms derived from relevance feedback. The reweighting of terms using feedback, although clearly a desirable technique, was beyond the scope of this experiment.

2.2. Methodology

The goal of this part of the experiment was to discover a method to filter the large number of additional terms provided by relevance feedback to a useful subset of about 20 terms for the interactive window. User interaction was simulated using the Cranfield 1400 test collection, with 1400 abstracts and 225 queries. Full words were used, and all queries were used in evaluation, not just those retrieving relevant documents on the first pass. The "frozen" method of evaluation was used [SALTON70], in which the top ten documents retrieved in the initial pass retain their ranks, making 11 the highest possible rank for the first feedback iteration. The methodology for feedback was as follows:

1) Run the best available ranking method for a query (see Appendix) and note which of the top ten documents retrieved for that query are relevant;

2) Create a file containing all non-common words from those relevant documents, including statistics concerning those terms;

3) Sort this term list based on a given statistical technique;

4) Add 20 terms from the top of the sorted list to the query;

5) Rerank the expanded query against the collection of unretrieved documents (no reweighting of query terms), and evaluate using the "frozen" method;

6) Repeat 3, 4, and 5 for each different statistical technique.

2.3. Results

The initial feedback results are shown in Table 1. Six different statistical techniques were used for sorting the list.

1) noise
2) postings
3) noise within postings
4) noise * frequency within postings
5) noise * frequency * postings
6) noise * frequency

The variable noise represents the given term's noise (see Appendix) based on the entire database, and the sort goes from lowest noise to highest noise. The variable postings is the number of postings within the set of relevant documents used for feedback, that is, the number of relevant documents within the top ten documents retrieved in the initial pass that contained the given term, and the sort goes from highest number of postings to lowest. Similarly the frequency is the total frequency of the given term within the set of relevant documents used for feedback, and the sort goes from highest total frequency to lowest.

TABLE 1

FEEDBACK PERFORMANCE USING CRANFIELD 225 AND ADDING TWENTY TERMS

		Variables For Sorting						
			1	2	3	4	5	6
		no feedback	noise	post	noise w/in post	noise*freq w/in post	noise*freq * post	noise*freq
A	% imp.avg. prec.	0	5.3	5.2	7.7	8.8	9.4	8.1
B	rel. by 10	650	650	650	650	650	650	650
	rel. by 20	830	905	909	945	954	962	951
	% improvement	0	41.7	43.9	63.9	68.9	73.3	67.2
	rel. by 30	946	1002	1031	1060	1076	1086	1076
	% improvement	0	18.9	28.7	38.5	43.9	47.3	43.9
C	queries imp.	0	70	82	94	94	91	92
	queries dec.	0	36	24	25	25	24	22

Three types of evaluation measures are shown in the table, (A) the average improvement in precision based on recall-level averages [SALTON83], (B) the counts of the number of relevant documents retrieved by a given document cutoff, and (C) the number of queries that show improvement or degradation in performance after thirty documents have been seen (20 previously unseen documents). As can be seen from the table, even the worst performance added 75 (905-830) more relevant documents by a cutoff of 20 documents retrieved than would have been retrieved had no terms been added, and the best performance added 132 more relevant, or an improvement of 73.3% over no feedback. This magnitude of performance improvement is not reflected in the change in average precision because of the requirement to freeze ranking for the initial retrieval. The improvement for feedback is clearly significant, with the best results showing performance improvement in 91 queries, a decrement in performance for only 24, and 110 queries with no change in performance.

The statistical technique used for sorting had a large effect on performance. Sorting the terms based strictly on their noise within the entire database (sort 1) performed the worst, indicating that the distribution of a term within the entire database is not very predictive of its value in query expansion. Using the number of postings of a term within the set of relevant documents used for feedback (sort 2) worked much better, as a term appearing in most of those relevant documents usually represents a concept central to those documents. However, the random ordering of the terms within a given number of postings is not as effective as a sort by noise within the number of postings (sort 3). The fourth and fifth sorts further refine the method by adding the effect of the frequency of the term within the set of relevant documents used for feedback. A term that appears frequently in a document is often an important term in that document, and this concept can be extended to a group of documents. In both sorts the frequency of the term within the set is used with a $\log_2$ function to dampen the effect. For example, the actual formula used for sort 5 is the noise of a term times the $\log_2$ of its frequency within the set times the number of postings within the set. Sort 6 eliminates the number of postings, and is not quite as effective, even though the frequency is related to the number of postings. As might be expected, there is significant performance improvement between the best sort, sort 5, and the worst sort, sort 1. However, although there appear to be differences in performance between the better sorts (sorts 3, 4, and 5), these are not significant based on a sign test using the number of queries that show improvements or degradations in performance.

2.4 Expanding the Number of Terms from Feedback

Whereas twenty terms seems to be an appropriate number of terms to provide for user selection, it is possible to show more than twenty terms to a user by employing a scrolling mechanism in the window. These additional terms must add significant performance improvement over the initial twenty terms, however, to justify the additional user effort. Experiments were run using the six sorting techniques and varying the number of terms added from only ten terms to forty terms. A sample of the results are shown in Figure 2, where the data from the worse sort (labeled "W") and the best sort (labeled "B") are plotted.

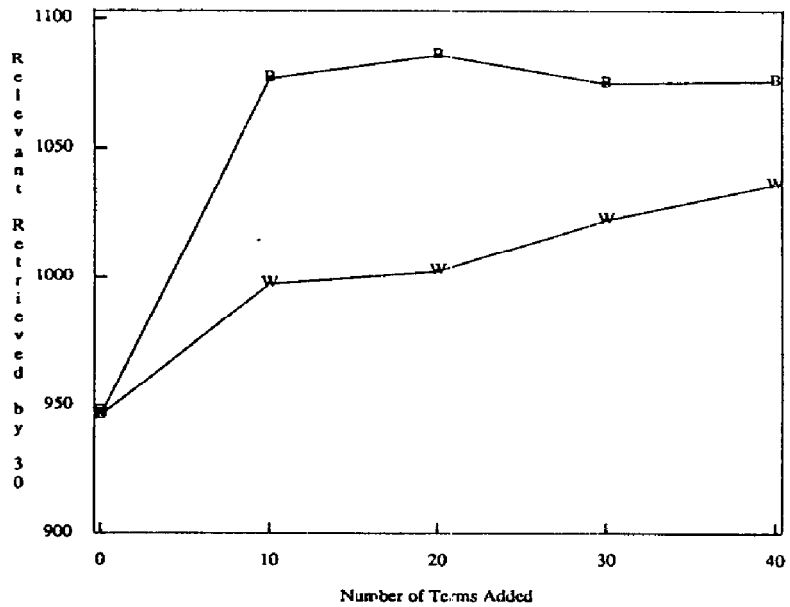

Figure 2 -- Feedback Performance as a Function of the Number of Terms Added to a Query

These results are typical for the other sorts in that the better sorts reached their peak performance after only twenty terms were added, with performance slowly decreasing after that. .The poorer sorts produced poor performance initially, but continued to improve as more terms were added, although never above the performance of the better sorts. This is a reflection on the effectiveness of the sorts in that the better sorts put the terms most useful for retrieval (for a given query) at the top of the list, and adding terms beyond the top twenty tended to add terms that were less useful.

The lack of improvement from adding more than twenty terms shows that the methods developed do act as an effective filter for the large number of terms provided by relevance feedback (an average of 162 per query), and suggests that previous lack of improvement from relevance feedback expansion may have been caused by too many terms being added to the query.

2.5 User Filtering

The large improvement in performance obtained by expanding the query by the top twenty terms in the sorted list measures the improvement if all terms from the window are added. In an interactive situation, the user will select only those terms deemed useful. An effort was made to see what further improvement this could bring by adding only those terms in the window that actually appear in relevant documents that have not been seen by the user. This represents a "perfect" choice by the user. The filtering reduced the set of twenty terms added to an average of 12 terms per query. The performance results were much improved, adding 97 more relevant documents to the total relevant retrieved by a cutoff of twenty documents, an improvement of 31% over performance with no user selection. User selection would not be perfect, of course, but most users will be able to eliminate many terms that are clearly unrelated to the query.

3. The Use of Suffixing to Produce Term Variants for Query Expansion (Window 2)

3.1 Past Work

The conflation of word variants using suffixing algorithms is one of the earliest enhancements to statistical keyword retrieval systems [SALTON68]. These systems automatically expand the query by its term variants, and use term weighting based on the root of the word rather than the word itself. Three stemming algorithms, the Lovins algorithm [LOVINS68], the Porter algorithm [PORTER80], and a singular-plural conflation algorithm, have become widely used in the retrieval community. In terms of retrieval performance, the Lovins and Porter algorithms were compared [LENNON81] using several collections, including the Cranfield collection (titles only) and were found to make very little difference in retrieval performance from that using full word retrieval. Further work using the Cranfield 1400 collection abstracts [HARMAN87] revealed that the addition of term variants from stem conflation does improve performance, but only for some queries. Overall performance on available test collections showed no significant improvement from the addition of term variants. A large number of queries have lower performance because many of the added term variants appear in no relevant documents for that query, causing a drop in precision. No technique was found for separating the useful term variants (for a given query) from the non-useful ones, and it was proposed that an interactive system might allow the user to provide the necessary separation.

3.1 *Methodology*

The goal of this part of the experiment was to discover a method to separate the useful term variants from the non-useful ones, or, failing in that, to estimate how much improvement can be gained from adding term variants that have been selected by the user. The methodology was as follows:

1) Run the best available ranking method for a query using full words;

2) Add all term variants of query terms to the query using the Lovins stemming algorithm;

3) Rerank the expanded query against the collection of unretrieved documents (no reweighting of query terms), and evaluate using the "frozen" method.

3.2 Results

The term variant expansion results are shown in Table 2.

TABLE 2
QUERY EXPANSION USING TERM VARIANTS, CRANFIELD 225

	Filtering Technique				
	no expansion	no filtering	feedback filtering	user filtering	feedback and user filtering
% imp.avg. prec.	0	-2.8	1.8	8.0	4.1
rel. by 10	650	650	650	650	650
rel. by 20	830	793	856	944	892
% improvement	0	-25.8	14.4	63.3	34.4
rel. by 30	946	832	973	1075	1007
% improvement	0	-27.6	9.1	43.6	20.6
queries imp.	0	33	40	91	50
queries dec.	0	77	32	17	18

The addition of term variants using the Lovins stemmer and no filtering produced a significant decrement in performance. This is consistent with past experience using stemmers, in that many of the term variants being added are not useful for retrieval (for the given query) and only cause precision drops. The performance decrement is somewhat smaller using the Porter algorithm or the singular-plural conflation algorithm (results not shown), but these stemmers add an average of only 17 and 5 terms respectively (for the Cranfield collection), and more possible expansion terms were needed for the window (Lovins adds an average of 39 terms per query).

Various filtering attempts were tried. Past work [HARMAN87] showed that filtering using the noise of a term in a database was not effective. The first new filtering technique tried was using the terms found in retrieved relevant documents (those retrieved in the first pass) as a filter on the term variants, i.e. only term variants that appear in those relevant documents are used for expansion. This proved to be a somewhat effective filter, producing significant performance improvement over no filtering, but minimal performance improvement over no query expansion by term variants.

User filtering, simulated in a similar manner to section 2.5, shows very significant performance improvement, however. This supports the proposal that a user could provide the necessary separation between useful and non-useful term variants, and additionally indicates that proper selection of term variants can produce performance improvements comparable to those seen using feedback. Again, whereas a user will not match the perfect selection performance, a substantial improvement should be possible.

A final experiment was run (feedback and user filtering) to test if a user should be shown the full set of Lovins term variants, or the feedback filtered set (an average of 3.2 terms per query). Clearly the full set is needed, as performance decreases significantly if a user is only given the limited set to select from.

4. The Use of Nearest Neighbor Routines for Query Expansion (Window 3)

4.1 Past Work

The use of thesauri requires the availability of a costly manually-built thesaurus, an option not normally available. Attempts have been made to construct variations on automatic thesauri, such as term-term correlation matrices, maximal spanning trees [HARPER78,VAN RIJSBERGEN81], or nearest neighbor expansions [WILLETT81], with some success. Smeaton [SMEATON82] did multiple experiments using the maximal spanning tree and the nearest neighbor techniques on the NPL test collection. Performance using the nearest neighbor expansion method was similar to that using the maximal spanning tree method, with little improvement noted for either. Smeaton suggested that the nearest neighbor to each term be added, rather than nearest neighbors to the group of terms in the query, and that not all terms need be expanded. These ideas were built on for the current work.

4.2 Methodology

The goal of this part of the experiment was to find effective nearest neighbor techniques to provide around twenty terms to fill window 3. The nearest neighbor algorithm described in [WILLETT81] was implemented, using the Dice coefficient as the similarity measure. This algorithm computes the relatedness of a given term to all other terms in the database, based on patterns of co-occurrance. A ranked list of related terms (nearest neighbors) is created, with the degree of relatedness calculated using the Dice coefficient. The methodology was as follows:

1) Run the best available-ranking method for a query using full words;

2) Calculate the nearest neighbors for each non-common term in the query;

3) Select some of the nearest neighbors from this list and add these terms to the query;

4) Rerank the expanded query against the collection of unretrieved documents (no reweighting of query terms), and evaluate using the "frozen" method.

4.3 Results

Four different selection techniques were tried, and the results are shown in Table 3.

TABLE 3
QUERY EXPANSION USING NEAREST NEIGHBORS, CRANFIELD 225

	Selection of Nearest Neighbors				
	no expansion	top 2 thres 8	top 2 thres 6	top 1 thres 6	top 5 thres 6 filtered
% imp.avg. prec.	0	-1.7	-0.9	-0.6	4.7
rel. by 10	650	650	650	650	650
rel. by 20	830	810	825	826	905
% improvement	0	-12.5	-2.9	-2.3	41.7
rel. by 30	946	900	918	929	1016
% improvement	0	-18.4	-10.4	-6.1	23.6
queries imp.	0	41	43	32	62
queries dec.	0	66	58	47	18

The first selection method uses the top 2 nearest neighbors for each term in the query that has a noise below a threshold of 8, which includes almost all terms except those with very high noise (nearly common terms). This selection method adds an average of 15 terms per query, and shows a decrement in performance from no expansion at all. The cause of the decrement is similar to the problem found when adding term variants, i.e., too many terms are being added to the query that are not useful for retrieval. The second selection technique adds fewer terms (an average of 8 per query) because query terms having a noise above 6 are not expanded. The results are better than the first technique, with fewer queries having a degradation in performance, but still no improvement over no query expansion. The third technique continues to use the lower noise threshold, but only adds the top nearest neighbor, cutting the number of terms being added to 4 on average. This further lowers the number of queries having a degradation in performance, but also lowers the number of queries showing improvement.

Clearly more terms need to be used for expansion, but not the set of terms selected by either the first technique or the second one. The final technique used added the top 5 nearest neighbors (an average of 20 new terms per query), but filtered these using the feedback terms, i.e. only those nearest neighbor terms that appear in the relevant documents retrieved on the first pass are used for expansion. This technique worked well for term variant filtering, and also works well here, showing significant improvement in performance, with 62 queries improving to only 18 queries showing a decrement in performance, compared to no query expansion.

A further set of experiments were run to determine how much improvement could be gained by user selection. Again hindsight was used to represent the "perfect choice" (as in section 2.5), and the results are shown in Table 4.

TABLE 4					
QUERY EXPANSION USING USER FILTERED NEAREST NEIGHBORS, CRANFIELD 225					
	Selection of Nearest Neighbors				
	no expansion	u.f. top 2 thres 6	u.f. top 1 thres 6	u.f. top 5 thres 6	u.f.top 5 thres 6 filtered
% imp.avg. prec.	0	6.0	3.6	8.7	7.3
rel. by 10	650	650	650	650	650
rel. by 20	830	911	889	957	938
% improvement	0	45.0	32.8	70.6	60.0
rel. by 30	946	1019	992	1079	1050
% improvement	0	24.6	15.5	44.9	35.1
queries imp.	0	68	47	95	77
queries dec.	0	21	17	19	14

As might be expected, the performance improvement was significant, with all user filtered nearest neighbor techniques performing significantly better than no expansion. The best performance was using the top 5 nearest neighbors, without the feedback filtering applied, but with user filtering. The user filtering reduced the average of 20 added terms per query to 4, and produced performance comparable to feedback expansion.

5. Combining Windows

The sample query screen shown in Figure 1 is the result of an actual expansion of a Cranfield query (query 2) using the best feedback sorting technique, the Lovins stemmer, and the top 5 nearest neighbors for all query terms with a noise below 6. Two facts are apparent from this example. First, most users can easily eliminate many words clearly of little use in retrieval for this query, such as "respect", "new", "discussed", "highly", "appears", "certain", and "having", along with technical terms not relevant to the query, such as "tools", "vtol", "fuselage", "aeroelastician" etc. This indicates that most users should experience performance closer to the "perfect choice" results shown for user filtering than to the nonfiltered results.

The second fact is the overlap of terms between tables. For example, the terms "structure" and "structures" appear in all three tables. Whereas the overlap is not high, the total performance using all three windows will be effected. Table 5 shows the effects of adding the term variants window to the feedback window, adding the nearest neighbor window to the feedback window, and finally using all three windows. The results are using the user filtered method in all cases.

TABLE 5							
QUERY EXPANSION COMBINING VARIOUS METHODS, CRANFIELD 225							
		Expansion method, user filtered					
	no expansion	feedback	term variant	nearest neighbor	feedback + term v	feedback + nn	feedback + term v + nn
% imp.avg. prec.	0	16.0	8.0	8.7	19.4	18.5	20.8
rel. by 10	650	650	650	650	650	650	650
rel. by 20	830	1059	944	957	1088	1093	1123
% improvement	0	127	63.3	70.6	143	146	163
rel. by 30	946	1165	1075	1079	1222	1213	1249
% improvement	0	122	91	95	93	90	102
queries imp.	0	122	91	95	147	146	156
queries dec.	0	11	17	19	10	9	8

As can be seen, using two types of query expansion together produces results significantly higher than a single method. In particular, adding user filtered term variants to user filtered feedback improves 25 more queries than user filtered feedback alone, and shows a 7% improvement in performance by a cutoff of twenty documents. Adding user filtered nearest neighbors to user filtered feedback improves 24 more queries, and shows an 8% improvement in performance by the same cutoff. Using all three windows improves 34 more queries than user filtered feedback alone, and shows a 16% improvement in performance by twenty documents retrieved. The total improvement over no query expansion, as shown in Table 5, is very substantial.

It is interesting to note that whereas the improvements to performance by a cutoff of twenty are relatively additive, the other evaluation measures indicate that combining all three methods of query expansion shows less improvement than might be expected. This seems to indicate that even though the terms being added are different, the same subset of documents are being retrieved. This subset of documents are those that are closely related to the query, but not closely enough related to be retrieved in the first pass. Possibly the 1249 relevant documents retrieved by rank 30 using this multi-window expansion method are nearly all that can be moved into high ranks by this technique. Other relevant documents either have poor abstracts containing few significant words, or are related to the query by some concept not easily translated into new query terms, at least not by these methods.

6. Conclusions

It has been shown that it is possible to generate short lists of terms that, when selectively added to a query, offer retrieval performance improvements that are very substantial. In particular, techniques have been developed to automatically select twenty feedback terms (from a set averaging around 160 in Cranfield) such that expanding the query by these terms provides significant performance improvement over no query expansion. It has further been shown that user selection from term variants and nearest neighbors of query terms can provide terms for query expansion that improve performance to that comparable with feedback. All three expansion techniques combined offer the user three lists of terms that can improve performance on the second pass by more than 160% over performance with no query expansion.

The costs of these techniques, both in terms of storage and response time, can range from minimal to substantial. The term variants and nearest neighbors of all database terms can be precalculated and stored in auxiliary files of sizes related to the number of unique terms in the database, usually a reasonable amount of storage, and the response time will not be significantly effected for users. The relevance feedback techniques, however, require assembling all terms contained in each retrieved relevant document, and this information is not easily available in most large-scale retrieval systems based on inverted files. Even if this information could be efficiently retrieved, the necessary retrieval and calculation to provide the merged, ranked term list could substantially effect response time. This additional cost remains a barrier to the use of relevance feedback for large-scale retrieval systems.

The techniques presented in this paper offer interactive retrieval systems the ability to provide constructive guidance to users during query modification. Even if no feedback were possible (due to either the high cost or the fact that no relevant documents were retrieved in the first pass), the term variant and nearest neighbor windows can be very effective in turning the query into a more productive set of terms. The use of short lists should encourage user participation, and the effectiveness of the new added terms should further this participation.

Acknowledgments

Thanks to the rest of my co-workers on the IRX team for providing a system so well adapted to research. In particular thanks to Rand Huntzinger for his help with UNIX, to Dennis Benson and Charles Goldstein for their helpful comments on this paper, and to Steve Pollitt for his stimulating discussions. Also thanks to former team member Larry Fitzpatrick for the nearest neighbor routine, and to the SMART project for the use of the evaluation and stemmer code.

REFERENCES

[CROFT83] Croft W.B., "Experiments with Representation in a Document Retrieval System", Information Technology: Research and Development 2 (1983), pp. 1-21.

[DENNIS64] Dennis S.F., "The Construction of a Thesaurus Automatically from a Sample of Text",Symposium Proceedings, Statistical Association Methods for Mechanized Documentation, 1964. (National Bureau of Standards Miscellaneous Publication 269).

[HARMAN86] Harman D., "An Experimental Study of Factors Important in Document Ranking", Proceedings of the 1986 ACM Conference on Research and Developments in Information Retrieval, Pisa, 1986.

[HARMAN87] Harman D., "A Failure Analysis on the Limitation of Suffixing in an Online Environment", Proceedings of the Tenth Annual International Conference on Research and Developments in Information Retrieval, New Orleans, 1987.

[HARPER78] Harper D.J. and van Rijsbergen C.J., "An Evaluation of Feedback in Document Retrieval using Co-Occurrence Data", Journal of Documentation, Vol. 34, No. 3, pp. 189-216, 1978.

[LENNON81] Lennon M., Peirce D., Tarry B. and Willett P., "An Evaluation of Some Conflation Algorithms for Information Retrieval", Journal of Information Science 3, pp. 177-188, 1981.

[LOVINS68] Lovins J.B., "Development of a Stemming Algorithm", Mechanical Translation and Computational Linguistics 11, March 1968, pp. 22-31.

[PORTER80] Porter M.F., "An Algorithm for Suffix Stripping", Program, Vol. 14, July 1980, pp. 130-137.

[PORTER82] Porter M.F., "Implementing a Probabilistic Information Retrieval System", Information Technology: Research and Development 1 (1982), pp. 131-156.

[ROBERTSON76] Robertson S.E. and Sparck Jones K., "Relevance Weighting of search terms", Journal of the ASIS, Vol. 27, No. 3, pp. 129-146, 1976.

[ROCCHIO68] Rocchio J.J., Jr., Relevance Feedback in Information Retrieval, Chapter 14 in [SALTON68].

[SALTON68] Salton G., The SMART Retrieval System--Experiments in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, N.J. 1968.

[SALTON70] Salton G., "Evaluation Problems in Interactive Information Retrieval", Information Storage Retrieval, Vol. 6, pp. 29-44, 1970.

[SALTON83] Salton G. and McGill M., Introduction to Modern Information Retrieval, McGraw-Hill Book Company, New York, 1983.

[SMEATON82] Smeaton A.F., The Retrieval Effects of Query Expansion on a Feedback Document Retrieval System, Technical Report 2, Department of Computer Science, University College Dublin (1982).

[SMEATON83] Smeaton A.F. and van Rijsbergen C.J., "The Retrieval Effects of Query Expansion on a Feedback Document Retrieval System", The Computer Journal, Vol. 26, pp. 239-246, 1983.

[SPARCK JONES72] Sparck Jones K., "A Statistical Interpretation of Term Specificity and Its Application in Retrieval", Journal of Documentation, Vol. 28, No. 1, March 1972, pp. 11-20.

[VAN RIJSBERGEN81] van Rijsbergen C.J., Harper D.J. and Porter M.F., "The Selection of Good Search Terms", Information Processing and Management, Vol. 17, pp. 77-91, 1981.

[WILLETT81] Willett P., "A Fast Procedure for the Calculation of Similarity Coefficients in Automatic Classification", Information Processing and Management, Vol. 17, pp. 53-60, 1981.

[WU81] Wu H. and Salton G., "The Estimation of Term Relevance using Relevance Feedback", Journal of Documentation, Vol. 37, No. 4, December 1981, pp. 194-214.

[YU76] Yu C.T. and Salton G., "Precision Weighting--An Effective Automatic Indexing Method", Journal of the ACM, Vol. 23, No. 1, PP. 76-88, 1976.

APPENDIX

$$\text{weight}_j = \sum_{k=1}^{Q} \frac{(\log_2 \text{Freq}_{jk} \times (\text{noise}_{max} - \text{noise}_k))}{\log_2 M}$$

where Q = the number of terms in the query
Freq_{jk} = the frequency of query term k in record j
noise_{max} = the maximum value of noise_k for a given database
M = the number of terms in the record j

$$\text{noise}_k = \sum_{i=1}^{N} \frac{\text{Freq}_{ik}}{\text{TFreq}_k} \log_2 \frac{\text{TFreq}_k}{\text{Freq}_{ik}}$$

where N = the number of records in the database
Freq_{ik} = the frequency of term k in record i
TFreq_k = the total frequency of term k in the database