Automatic Query Expansion Using SMART : TREC 3

Chris Buckley*, Gerard Salton, James Allan, Amit Singhal

Abstract

The Smart information retrieval project emphasizes completely automatic approaches to the understanding and retrieval of large quantities of text. We continue our work in TREC 3, performing runs in the routing, ad-hoc, and foreign language environments. Our major focus is massive query expansion: adding from 300 to 530 terms to each query. These terms come from known relevant documents in the case of routing, and from just the top retrieved documents in the case of ad-hoc and Spanish. This approach improves effectiveness from 7% to 25% in the various experiments.

Other ad-hoc work extends our investigations into combining global similarities, giving an overall indication of how a document matches a query, with local similarities identifying a smaller part of the document which matches the query. Using an overlapping text window definition of "local", we achieve a 16% improvement.

Introduction

For over 30 years, the Smart project at Cornell University has been interested in the analysis, search, and retrieval of heterogeneous text databases, where the vocabulary is allowed to vary widely, and the subject matter is unrestricted. Such databases may include newspaper articles, newswire dispatches, textbooks, dictionaries, encyclopedias, manuals, magazine articles, and so on. The usual text analysis and text indexing approaches that are based on the use of thesauruses and other vocabulary control devices are difficult to apply in unrestricted text environments, because the word meanings are not stable in such circumstances and the interpretation varies depending on context. The applicability of more complex text analysis systems that are based on the construction of knowledge bases covering the detailed structure of particular subject areas, together with inference rules designed to derive relationships between the relevant concepts, is even more questionable in such cases. Complete theories of knowledge representation do not exist, and it is unclear what concepts, concept relationships, and inference rules may be needed to understand particular texts.[13]

Accordingly, a text analysis and retrieval component must necessarily be based primarily on a study of the available texts themselves. Fortunately very large text databases are now available in machine-readable form, and a substantial amount of information is automatically derivable about the occurrence properties of words and expressions in natural-language texts, and about the contexts in which the words are used. This information can help in determining whether a query and a text are semantically homogeneous, that is, whether they cover similar subject areas. When that is the case, the text can be retrieved in response to the query.

Automatic Indexing

In the Smart system, the vector-processing model of retrieval is used to transform both the available information requests as well as the stored documents into vectors of the form:

$$D_i = (w_{i1}, w_{i2}, \dots, w_{it})$$

where D_i represents a document (or query) text and w_{ik} is the weight of term T_k in document D_i. A weight of zero is used for terms that are absent from a particular document, and positive weights characterize

*Department of Computer Science, Cornell University, Ithaca, NY 14853-7501. This study was supported in part by the National Science Foundation under grant IRI 93-00124.

terms actually assigned. The assumption is that t terms in all are available for the representation of the information.

In choosing a term weighting system, low weights should be assigned to high-frequency terms that occur in many documents of a collection, and high weights to terms that are important in particular documents but unimportant in the remainder of the collection. The weight of terms that occur rarely in a collection is relatively unimportant, because such terms contribute little to the needed similarity computation between different texts.

A well-known term weighting system following that prescription assigns weight w_{ik} to term T_k in query Q_i in proportion to the frequency of occurrence of the term in Q_i, and in inverse proportion to 'the number of documents to which the term is assigned.[14, 12] Such a weighting system is known as a tf×idf (term frequency times inverse document frequency) weighting system. In practice the query lengths, and hence the number of non-zero term weights assigned to a query, varies widely. To allow a meaningful final retrieval similarity, it is convenient to use a length normalization factor as part of the term weighting formula. A high-quality term weighting formula for w_{ik}, the weight of term T_k in query Q_i is

$$w_{ik} = \frac{(\log(f_{ik}) + 1.0) * \log(N/n_k)}{\sqrt{\sum_{j=1}^{t} [(\log(f_{ij}) + 1.0) * \log(N/n_j)]^2}} \tag{1}$$

where f_{ik} is the occurrence frequency of T_k in Q_i, N is the collection size, and n_k the number of documents with term T_k assigned. The factor $\log(N/n_k)$ is an inverse collection frequency ("idf") factor which decreases as terms are used widely in a collection, and the denominator in expression (1) is used for weight normalization. This particular form will be called "ltc" weighting within this paper.

The weights assigned to terms in *documents* are much the same. In practice, for both effectiveness and efficiency reasons the *idf* factor in the documents is dropped.[2, 1]

The terms T_k included in a given vector can in principle represent any entities assigned to a document for content identification. In the Smart context, such terms are derived by a text transformation of the following kind:[12]

1. recognize individual text words
2. use a stop list to eliminate unwanted function words
3. perform suffix removal to generate word stems
4. optionally use term grouping methods based on statistical word co-occurrence or word adjacency computations to form term phrases (alternatively syntactic analysis computations can be used)
5. assign term weights to all remaining word stems and/or phrase stems to form the term vector for all information items.

Once term vectors are available for all information items, all subsequent processing is based on term vector manipulations.

The fact that the indexing of both documents and queries is completely automatic means that the results obtained are reasonably collection independent and should be valid across a wide range of collections. No human expertise in the subject matter is required for either the initial collection creation, or the actual query formulation.

Phrases

The same phrase strategy (and phrases) used in TREC 1 and TREC 2 ([2, 1]) are used for TREC 3. Any pair of adjacent non-stopwords is regarded as a potential phrase. The final list of phrases is composed of those pairs of words occurring in 25 or more documents of the initial TREC 1 document set. Phrase weighting is again a hybrid scheme where phrases are weighted with the same scheme as single terms, except that normalization of the entire vector is done by dividing by the length of the single term sub-vector only. In this way, the similarity contribution of the single terms is independent of the quantity or quality of the phrases.

Text Similarity Computation

When the text of document D_i is represented by a vectors of the form $(d_{i1}, d_{i2}, \ldots, d_{it})$ and query Q_j by the vector $(q_{j1}, q_{j2}, \ldots, q_{jt})$, a similarity (S) computation between the two items can conveniently be obtained as the inner product between corresponding weighted term vector as follows:

$$S(D_i, Q_j) = \sum_{k=1}^{t} (d_{ik} * q_{jk}) \tag{2}$$

Thus, the similarity between two texts (whether query or document) depends on the weights of coinciding terms in the two vectors.

TREC 3 Approaches

One way to improve effectiveness is to better represent the information need by adding useful terms to the query. The classical example of this is relevance feedback, where terms occurring in known relevant documents are added to the query.

The relevance feedback process can be divided into two phases: query term selection and query term weighting. Our basic approach to relevance feedback heavily emphasizes query term weighting. Proper weighting allows us to massively expand the query by adding any term for which we have any evidence of usefulness. Experiments show that effectiveness improves linearly as the log of the number of added terms, up to a point of diminishing improvement [3]. This point of diminishing returns for the TREC environment seems to be about 300 terms.

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 expansion establishes a background "noise" similarity due to random poor term matches. The good documents escape the noise due to several good terms co-occurring within the document.

Some other expansion methods (eg naive thesaurus lookup) do not share the above property. When expansion occurs inappropriately, several connected poor words are added. Attempting to expand the word "bank" for instance, one might add several financial terms, which may reinforce each other and cause financial documents to be retrieved. This would result in poor retrieval if "bank" were referring to the side of a river. The poor terms for this query expansion are not co-occurring randomly and have a much greater effect on the final similarity.

Expansion by hundreds of terms occurring in known relevant documents worked so successfully in routing of TREC 2, that it was decided to use the same expansion techniques in the ad-hoc portion of TREC 3. In the ad-hoc environment, there are no known relevant documents. Instead, the top retrieved documents are all assumed to be relevant for the purposes of expansion and weighting. If many of the top documents are relevant, then the process achieves the same effect as relevance feedback. If none of the top documents are relevant, then the expansion is likely to have a very negative effect as the refashioned query will emphasize the same mistakes that caused the poor initial retrieval. The end result is thus likely to be a mixture of improvements for many queries, but deterioration of results for others.

The idea of treating the top documents as being relevant in the absence of any real relevance judgements is not a new one. It has probably been done dozens of times in the past (eg, it was a standard Cornell information retrieval class project in the early 1980's). In general, at least in the Cornell experience, it helped some queries but the negative results predominated on the standard small test collections. What makes the approach successful in the TREC environment is the combination of better initial retrieval, and the collection characteristics of TREC. There are many more relevant documents per query within TREC, and those documents are longer than in the small test collections. So there is more of a chance for terms from the relevant retrieved documents to meaningfully distinguish themselves from the terms in the non-relevant documents. Other groups in TREC 2 were able to take advantage of this situation and improve performance, noticeably UCLA and CMU [6, 7].

Another focus of our work the past few years, both within TREC and outside it, has been trying to take advantage of local similarities between a small part of the document and the query. We've shown that local similarities can be used very effectively to ensure that terms in common between document and query are being used in the same semantic sense. However, while this semantic disambiguation is important in other environments [16], the very long and rich TREC queries provide enough global context to disambiguate without going to a local level. Our efforts to improve effectiveness using local disambiguation using sentences and short paragraphs did not work in TREC 1 and TREC 2. For TREC 3, we lengthen our local contexts, and treat the local passage as being a mini-document. Adopting the approach of UMass [17, 4, 5], we define our local contexts to be a set of overlapping text windows, each of fixed size. This avoids the length and normalization problems that adversely affected our approach in TREC 2.

System Description

The Cornell TREC experiments use the SMART Information Retrieval System, Version 11, and are run on a dedicated Sun Sparc 20/51 with 160 Megabytes of memory and 18 Gigabytes of local disk.

SMART Version 11 is the latest in a long line of experimental information retrieval systems, dating back over 30 years, developed under the guidance of G. Salton. Version 11 is a reasonably complete re-write of earlier versions, and was designed and implemented primarily by C. Buckley. The new version is approximately 44,000 lines of C code and documentation.

SMART Version 11 offers a basic framework for investigations of the vector space and related models of information retrieval. Documents are fully automatically indexed, with each document representation being a weighted vector of concepts, the weight indicating the importance of a concept to that particular document (as described above). The document representatives are stored on disk as an inverted file. Natural language queries undergo the same indexing process. The query representative vector is then compared with the indexed document representatives to arrive at a similarity (equation (2)), and the documents are then fully ranked by similarity.

Routing Experiments

Our routing experiments in TREC 3 are only slightly different from those carried out for TREC 2. The basic routing approach chosen is the feedback approach of Rocchio [11, 15, 1]. Expressed in vector space terms, the final query vector is the initial query vector moved toward the centroid of the relevant documents, and away from the centroid of the non-relevant documents.

$$\begin{aligned} Q_{\text{new}} &= A * Q_{\text{old}} \\ &+ B * \text{average\_wt\_in\_rel\_docs} \\ &- C * \text{average\_wt\_nonrel\_docs} \end{aligned}$$

Terms that end up with negative weights are dropped (less than 3% of terms were dropped in the most massive query expansion below).

The parameters of Rocchio's method are the relative importance of the original query, the relevant documents, and the non-relevant documents (A,B,C above); and then exactly which terms are to be considered part of the final vector.

The investigations of TREC 2 and elsewhere [3] suggest that the decision of which terms to add is not a hard decision: just add all terms occurring in relevant documents that can be efficiently handled. We sort all terms occurring in the relevant documents by the number of relevant documents in which they occur, with ties being broken by considering the highest average weight in the relevant documents. We then add the top 300 single terms and top 30 phrases to the original query and reweight according to the Rocchio formula above with A,B,C parameters being 8,16,4. This forms the queries for our CrnlRR run.

Query-by-Query Variations

While the massive expansion Rocchio approach works well for most queries, examining past individual query results reveals that for about 15% of the queries, not expanding works better than massive expansion[3].

Run	Best	$\geq$ median	$<$ median
CrnlRR	1	44	5
CrnlQR	2	37	11

Table 1: Comparative Routing Results

	Run	$X.Y$	$A.B.C$	R-prec	Total Rel	recall-prec
1.	no fdbk	0.0	8.0.0	3461	5975	2985
2.	no expand	0.0	8.8.4	3698	6342	3163
3.	CrnlRR	300.30	8.16.4	4064	7134	3699
4.	CrnlQR	varies	varies	4013	7215	3725

Table 2: Routing evaluation

Our second official run, like our second official run of TREC 2, is an attempt to choose feedback parameters on a per-query basis to avoid expanding on those queries where no expansion might be appropriate. We also want to examine other feedback approaches for those queries with no, or little, expansion. In TREC 1 and TREC 2 it was noticed that the probabilistic approaches, e.g., the classical probabilistic formula [10] and Dortmund's RPI formula [8, 9], did better than the Rocchio approach if there was little expansion. Perhaps a choice among feedback methods would improve effectiveness; our TREC 2 results suggested just changing expansion amounts on a per-query basis would yield only a small improvement.

We examine seven different approaches:

1. : Original query, no expansion or reweighting.
2. : Probabilistic weights, no expansion.
3. : RPI model, no expansion
4. : RPI model, expansion by 30 single terms
5. : Rocchio, expansion by 30 single terms and 10 phrases
6. : Rocchio, expansion by 500 single terms
7. : Rocchio, expansion by 500 single terms and 30 phrases, with A,B,C parameters being 8,32,4

We ran each of these approaches using the 50 queries of the TREC 3 routing task, learning on D1 and testing on D2. This determined which approach should be used for which queries in the routing task. The final queries for the CrnlQR run are formed by using the best approach on each query, and learning from the full D12 set of relevance judgements.

Routing Results

Both CrnlRR and CrnlQR do quite well in comparison with other TREC 3 routing runs (Table 1). These comparative results are not quite as good as in TREC 2, suggesting that some other groups might have caught up to us.

Evaluation measures in Table 2 for both the official and some non-official runs show the importance of query expansion. Run 1 is the base case original query only (ltc weights). Just re-weighting the query terms without adding any terms according to Rocchio's algorithm gives a 6% improvement. Both reweighting and massively expanding gives a 24% improvement.

The run CrnlQR is actually quite disappointing. Like our initial attempts at per-query variations in TREC 2, we get very little improvement over using massive expansion for all queries. Table 3 shows that

	Approach	Expansion Sing.phrs	Rocchio $A.B.C$	Num Queries	Num better than CrnlRR
1.	Orig. query	0.0	n.a.	3	1
2.	Prob	0.0	n.a.	4	0
3.	RPI	0.0	n.a.	9	3
4.	RPI	30.0	n.a.	2	1
5.	Rocchio	30.10	8.16.4	3	1
6.	Rocchio	500.0	8.16.4	9	5
7.	Rocchio	500.30	8.32.4	20	15

Table 3: Routing Approach Variation

of all the feedback variations tried, the only ones that consistently do better than the CrnlRR Rocchio expansion by 330 terms, are the queries which use Rocchio and expand by even more terms. The low expansion approaches did better on their queries when learning on D1 and testing on D2, but tended to do worse on their queries when learning on D12 and testing on D3. This suggests that there is no inherent property of the semantics of an individual query that can predict whether massive expansion will work. Instead, it suggests the effectiveness of massive expansion depends on the properties of the documents, in both the learning set and the test set.

Ad-hoc Results

The first of Cornell's two ad-hoc runs, CrnlEA, is very similar to the Rocchio routing run, CrnlRR. The initial query is expanded and reweighted using Rocchio's feedback approach. The major difference is that there are no known relevant documents from which to draw the expansion terms. Instead, an initial retrieval is done, and the top 30 documents are all assumed to be relevant for the purposes of expansion and reweighting. While this is certainly not as good as having real relevance judgements (especially if the initial retrieval obtains no relevant documents), these terms should still have some connection to relevance.

The initial query is expanded by 500 terms and 10 phrases. In the future, perhaps more phrases should be chosen. However, in this initial experiment having many phrases would complicate the analysis of what is actually happening. The $A.B.C$ parameters of the Rocchio equation are set to 8.8.0. These parameters weight the original query terms higher than in standard relevance feedback, and disregard occurrences among the non-relevant documents. The parameters were chosen after a small set of trial runs using the first 150 queries on D12.

The second of Cornell's ad-hoc runs, CrnlLA, is this year's local/global run. At retrieval time, each document is assigned a similarity based upon both the document's global similarity to the query, and upon the similarities of smaller parts of the document to the query. For this experiment, the parts of the document are defined to be text windows of 200 words in length. One set of text windows starts at the beginning of the document, with a new window every 200 words. Another set of text windows on that document starts 100 words into the document, with a new window every 200 words. The two sets of overlapping windows ensure that every semantically coherent chunk of text of length less than 100 words will be included whole in at least one text window.

The text of each local window is indexed and weighted with binary term weights (SMART-nomenclature "bnn" weights). The weights in the text windows do not need to be normalized since the text windows are almost all of the same length. An "idf" factor does not need to be included in the local document weights since it will be included in the query weight of any matching term. A pure "tf" factor that gives a weight proportional to the number of times a term occurs in the text window will over-weight common words. Thus the "bnn" weighting scheme would seem to be appropriate.

The question of how to combine a global similarity with the local similarity of a document has yet to be resolved. Work done in preparation for TREC 2 strongly suggested that the result should be some combination of the global similarity with the best local similarity of the document (as opposed to, say, an

Run	Best	$\geq$ median	$<$ median
CrnlEA	3	38	9
CrnlLA	0	49	1

Table 4: Comparative Ad-hoc results

average of local similarities). Other work showed that the values of both global and local similarities are query dependent. A good local similarity for one query may be a poor local similarity for another query. This suggests some sort of query relativization factor may be needed. Several functions were tried in preparation for TREC 3; the one used for the run CrnlLA is

$$\text{FinalSim} = \text{GlobalSim} + 2 * \text{GlobalSim} * \text{LocalSim}/\text{BestLocalSim} \tag{3}$$

where

- GlobalSim is the global similarity between an "ltc" weighted query, Q, and an "lnc" weighted document, D.
- LocalSim is the highest similarity of any "bnn" weighted text window of D with the "ltc" weighted query Q.
- BestLocalSim is the highest LocalSim for any examined document for this query Q.

The CrnlLA retrieval procedure to return rankings for 1000 documents is to

1. Perform a global search retrieving the top 1750 documents.
2. For each retrieved document,
   (a) Fetch the original document,
   (b) Break it into text windows,
   (c) Index and weight each text window separately
   (d) Calculate the similarity of each text window to the query.
   (e) Set the document's LocalSim to the highest of these similarities.
3. Set BestLocalSim to the highest LocalSim among the 1750 documents
4. Use Equation 3 to calculate a final similarity for the 1750 documents.
5. Rank the final similarities and return the top 1000.

The expansion run, CrnlEA, and the local/global run, CrnlLA, are very different but each perform well when compared against other systems. Both approaches perform at or above the median in most queries, as can be seen in in Table 4.

As could be expected, CrnlEA is somewhat inconsistent, performing extremely well on some queries, but dipping below the median on several others. Presumably this is related to the quality of the initial search, though this has not yet been tested. CrnlLA is almost always above the median, but was never the highest rated run.

Table 5 gives the results of several evaluation measures for CrnlEA, CrnlLA, and a simple "lnc.ltc" vector run. Each of the TREC 3 approaches gives substantial recall-precision improvement over the pure vector run (20.3% for CrnlEA, and 16.2% for CrnlLA). However, they get this improvement in very different fashions.

Run	Recall-Precision	Total Rel Retrieved	Precision 5 docs	Precision 100 docs
lnc.ltc	2842	6531	5530	3780
CrnlEA	3419	7267	5760	4168
CrnlLA	3302	6808	6800	4216

Table 5: Ad-hoc results

CrnlEA is a recall oriented approach. It shows a very mild .023 improvement in precision at 5 documents, but retrieves a very strong 736 more relevant documents than the vector run. CrnlLA, on the other hand, is a precision oriented approach. It shows a very strong .1270 improvement in precision at 5 documents, but then a much weaker increase of 277 relevant documents retrieved. It remains to be seen whether the strengths of these two very different approaches can be combined in one run.

Spanish

One of the fun side-tracks of TREC 3 is the Spanish experiments. About 200 megabytes of Spanish text and 25 Spanish queries were made available for runs in the ad-hoc environment. Our claim has always been that SMART is to a large extent language independent, as long as the language is based upon recognizable word tokens. TREC 3 Spanish presented a chance to test this claim.

Spanish SMART

Unlike other retrieval systems, SMART uses almost no linguistic knowledge. Enabling SMART to run well on Spanish text only required 3 subtasks.

1. Make SMART 8-bit clean.
2. Fashion stemming rules for Spanish.
3. Construct a stopword list of common Spanish words.

Extending SMART to handle 8-bit characters (e.g., the accented Spanish characters) instead of 7-bit ASCII was very simple. About 8 lines of code needed changing, plus a 128 entry table in the tokenizer giving the class of characters needed to be expanded to 256 entries.

After this was done, the Spanish document set was indexed without any stemming rules or stopwords. Simple stemming rules were then derived by looking at the sorted dictionary entries and guessing which lexicographically adjacent entries really represented the same words (guessing since the person doing this does not speak Spanish!). The final stemming rules were:

- Remove final “as”, “es”, “os”, “a”, “o”, “e”.
- Change final “z” to “c”.

Initially the stopword list was composed of the 800 most frequently occurring words in the collection. This was later trimmed to 342 words by asking a native Spanish speaker to prune the list.

The Spanish collection was then re-indexed using the new stemming rules and stopword list, and was ready for use. It was, however, somewhat disconcerting to type the first query “This is a test”, and retrieve a large set of English documents dealing with standard tests! The explanation turned out to be a partial file of English documents that had somehow crept into the distributed collection.

The total time to make SMART Spanish ready was about 5-6 person-hours.

Run	Best	≥ median	< median
CrnlES	11	8	4
CrnlVS	1	13	9

Table 6: Comparative Spanish Ad-hoc results

Run	Recall-Precision	Total Rel Retrieved	Precision 5 docs	R-Precision
CrnlES	5692	2439	7917	5578
CrnlVS	5301	2402	7500	5328

Table 7: Spanish Ad-hoc results

Spanish Ad-Hoc Runs

The two Cornell Spanish runs are CrnlVS, a simple "lnc.ltc" vector run, and CrnlES, a massive expansion run. Both procedures are described above in the main-line ad-hoc description; aside from the different database names there is no difference in the scripts which run the experiments.

Table 6 shows the two runs both do very well, though the expanded run is significantly better. CrnlES has the best results on 11 out of the 23 Spanish queries with relevance documents. (But remember that many fewer groups submitted Spanish runs, so our "share" of best results is expected to be higher.)

The results of the standard evaluation measures show extremely good retrieval effectiveness in Table 7. However, many of these values are artificially high. Unlike the mainstream ad-hoc and routing pools of judged documents, the Spanish pool was very small and narrow, and it's clear that a lower percentage of relevant documents were judged, thus somewhat inflating the recall figures. Comparative evaluation between runs should still be valid though, even between judged and unjudged runs. For example, CrnlES is definitely better than CrnlVS even though CrnlVS was a judged run and CrnlES was not.

Efficiency

Efficiency issues are becoming increasingly important in these TREC experiments as retrieval methods become more complicated and expensive. Thus it is important to have at least some discussion of efficiency within a paper like this.

SMART is a reasonably fast system. It indexes documents at a rate of about 600 megabytes per hour. Simple vector retrieval runs can be quite fast. Calculating the similarities for the CrnlVS run took much less than 2 seconds for all 25 queries together (keeping track of the top 1000 documents for each query was done rather inefficiently and took much longer!). But the more complicated retrieval methods take anywhere from 9 seconds per query (CrnlRR) to 189 seconds per query (CrnlLA).

Luckily, in actual practice the execution times of the complicated methods can be cut down drastically. The massive query expansion approaches will benefit greatly from optimization efforts such as those discussed in our TREC 1 work. Some of the effectiveness increase of the massive query expansion will have to be traded back in order to get reasonable efficiency, but the results of TREC 1 show the effectiveness cost will not be prohibitive.

The other very time consuming approach of ours is the local/global matching (CrnlLA). The re-indexing of the local parts of a document can be done off-line and stored. When this time savings is combined with the decreased time due to a user asking for a reasonable number of documents (instead of 1000), retrieval time should be not much more than double an ordinary vector search. This should be quite feasible in practice, depending on the particular constraints of a site, of course.

Conclusion

Automatic massive query expansion proves to be very effective for routing. Conventional relevance feedback techniques are used to weight the expanded queries. Once again, however, the option to choose feedback approaches on a per-query basis doesn't help significantly, where the choice is based on what worked in the past for this query.

Massive query expansion also works in general for the ad-hoc experiments, where expansion and weighting are based on the top initially retrieved documents instead of known relevant documents. In the ad-hoc environment this approach may hurt performance for some queries (e.g., those without many relevant documents in the top retrieved set), but overall proves to be worthwhile with an average 20% improvement.

Incorporating both global and local similarity information in the final ranking is useful, with improvements of 16%. Care needs to be taken, though, both in the definition of a local part of a document (making all parts equal length helps the weighting task enormously), and in the combination of the local and global similarity.

SMART is very easily adaptable to at least some foreign languages, even without knowledge of the languages. Performance of SMART appears to be just as good in the foreign language as in English, though this is tough to judge.

References

[1] Chris Buckley, James Allan, and Gerard Salton. Automatic routing and ad-hoc retrieval using SMART : TREC 2. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 45–56. NIST Special Publication 500-215, March 1994.

[2] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using SMART. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72. NIST Special Publication 500-207, March 1993.

[3] Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In W. Bruce Croft and C.J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300, New York, July 1994. Springer-Verlag.

[4] James P. Callan and W. Bruce Croft. An evaluation of query processing strategies using the TIPSTER collection. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–355, New York, June 1993. Association for Computer Machinery.

[5] W. Bruce Croft, James Callan, and John Broglio. TREC-2 routing and ad-hoc retrieval evaluation using the INQUERY system. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 75–84. NIST Special Publication 500-215, March 1994.

[6] E. Efthimiadis and P. Biron. UCLA-Okapi at TREC-2: Query expansion experiments. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 279–290. NIST Special Publication 500-215, March 1994.

[7] D. Evans and R. Lefferts. Design and evaluation of the CLARIT-TREC-2 system. In D. K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 137–150. NIST Special Publication 500-215, March 1994.

[8] Norbert Fuhr. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.

[9] Norbert Fuhr and Chris Buckley. Optimizing document indexing and search term weighting based on probabilistic models. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 89–99. NIST Special Publication 500-207, March 1993.

[10] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[11] J.J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, 1971.

[12] Gerard Salton. *Automatic Text Processing — the Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley Publishing Co., Reading, MA, 1989.

[13] Gerard Salton. Developments in automatic text retrieval. *Science*, 253:974–980, August 1991.

[14] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[15] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[16] Gerard Salton and Chris Buckley. Automatic text structuring and retrieval: Experiments in automatic encyclopedia searching. In *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–30, 1991.

[17] Craig Stanfill and David L. Waltz. Statistical methods, artificial intelligence, and information retrieval. In Paul S. Jacobs, editor, *Text-based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1971.