Using Clustering and SuperConcepts Within SMART : TREC 6

Chris Buckley*, Mandar Mitra†, Janet Walz*, Claire Cardie†

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 6, performing runs in the routing, ad-hoc, and foreign language environments, including cross-lingual runs. The major focus for TREC 6 is on trying to maintain the balance of the query – attempting to ensure the various aspects of the original query are appropriately addressed, especially while adding expansion terms. Exactly the same procedure is used for foreign language environments as for English; our tenet is that good information retrieval techniques are more powerful than linguistic knowledge. We also give an interesting cross-lingual run, assuming that French and English are closely enough related so that a query in one language can be run directly on a collection in the other language by just "correcting" the spelling of the query words. This is quite successful for most queries.

Introduction

For over 30 years, the Smart project at Cornell University, under the direction of the late Gerry Salton, has been investigating the analysis, search, and retrieval of heterogeneous text databases, where the vocabulary is allowed to vary widely, and the subject matter is unrestricted. Our belief is that text analysis and retrieval must necessarily be based primarily on a study of the available texts themselves. The community does not understand natural language well enough at the present time to make use of a more complex text analysis. Knowledge bases covering the detailed structure of particular subject areas, together with inference rules designed to derive relationships between the relevant concepts, are very difficult to construct, and have not yet been proven to aid in general retrieval.

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:

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

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

*SabIR Research

†Department of Computer Science, Cornell University, Ithaca, NY 14853-7501

The basic "tf*idf" weighting schemes used within SMART have been discussed many times. For TREC 6 we use the same basic weights and document length normalization as were developed at Cornell by Amit Singhal for TREC 4([4, 13]. Tests on various collections show that this indexing is reasonably collection independent and thus should be valid across a wide range of new collections. No human expertise in the subject matter is required for either the initial collection creation, or the actual query formulation.

The same phrase strategy (and phrases) used in all previous TRECs ([2, 1, 3, 4, 5]) are used for TREC 6. 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. Phrases are weighted with the same scheme as single terms.

Text Similarity Computation

When the text of document D_i is represented by a vector 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 vectors as follows:

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

Thus, the similarity between two texts (whether query or document) depends on the weights of coinciding terms in the two vectors. The SuperConcept similarity function described later will be a slight variant of this inner product function, but still depends on weights of coinciding terms.

System Description

The Cornell TREC experiments use the SMART Information Retrieval System, Version 13, and most were run on a dedicated Sun Sparc 20/51 with 160 Megabytes of memory and 33 Gigabytes of local disk (some supporting runs were made on a Sun UltraSparc 1/140).

SMART Version 13 is the latest in a long line of experimental information retrieval systems, dating back over 30 years, developed under the guidance of G. Salton. The new version is approximately 44,000 lines of C code and documentation.

SMART Version 13 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 (1)), and the documents are then fully ranked by similarity.

SMART is highly flexible and very fast, thus providing an ideal platform for information retrieval experimentation. Documents for TREC 6 are indexed at a rate of over a Gigabyte an hour, on hardware costing under $10,000 new. Retrieval speed is similarly fast, with basic simple searches taking much less than a second a query.

Ad-hoc

Ad-hoc Methodology

Automatic query expansion using pseudo relevance feedback has proven itself to be very useful in past TREC's ([7, 8, 3, 12]. In this approach:

1. A set of documents is initially retrieved in response to a user query

2. The top-ranked documents are assumed to be relevant (without any intervention from the user)
3. Low-ranked documents are optionally assumed to be non-relevant
4. These documents are used in a traditional Rocchio feedback method to expand and reweight the query
5. This new query is run against the document collection, providing the final retrieved set to be evaluated.

It is important to reinforce that this approach is entirely automatic; there is no user intervention or choice of terms or documents.

There are two steps in the automatic query expansion procedure that we can improve. First, the initial retrieval can be improved so that the assumption of relevance for the top-ranked documents is more accurate. Secondly, the expansion procedure can be improved to yield a better final query. For our TREC 6 ad-hoc runs, we attempt to improve both these steps.

Better initial retrieval. The query coverage (QC) algorithm used in our TREC 5 ad-hoc run is reused for the TREC 6 task to improve the set of documents assumed relevant. Briefly, this algorithm retrieves a few (50, say) documents using the simple vector similarity and computes a new similarity between queries and top-ranked documents based on whether query terms occur close to each other in a document (within a window of 50 terms, say), and whether the matching query terms are "independent" (in the sense that their occurrences are uncorrelated, i.e., the occurrence of one term is not a reasonable predictor of the presence of another). The top 50 documents are re-ranked based on this new similarity and the top 20 in the resulting ranking are assumed relevant. Using this refined set of 20 documents in the feedback process yielded good improvements at TREC 5 [5].

We also experimented with other techniques to improve the initial retrieval. We used natural language processing techniques to identify phrases in queries and documents, hoping that the high-quality phrases identified this way could be used to better predict the relevance of a document. We found however that phrases have little effect on the ordering of documents at top ranks and thus cannot be used to significantly improve the quality of the initially retrieved set. (The details of our experiments are reported in [10].) We therefore did not use this method in our official TREC 6 submissions.

Another approach involved the use of Boolean filters to refine the initially retrieved set (see Hearst in [9]). By filtering out top-ranked documents that fail to satisfy certain Boolean constraints, we can eliminate several non-relevant documents and increase the proportion of relevant documents at top ranks. Initial experiments using manually formulated filters yielded some improvement, but did not outperform the automatic QC algorithm described above. Further, the automatic generation of appropriate Boolean filters given a natural language query is a difficult task, and automatic filters are expected to do worse than the manual filters used in our experiments. Thus, this approach was also not pursued for our TREC 6 run, though we have continued our research on it ([11]).

Improved expansion. We tried using document clustering as well as term clustering to improve the expansion step. These approaches were also motivated by query coverage — we would like to ensure that multiple key concepts in the user query are nicely represented in a balanced way in the expanded query.

When the top-ranked documents for a query are clustered, the different clusters often correspond to different key concepts. For example, for TREC query 203 (*What is the economic impact of recycling tires?*), the top documents could form clusters corresponding to *recycling* (discussing garbage, recycling of plastic, tin cans, etc.), *tires* (dealing with rubber, automobile tires, the Goodyear company, etc.), and *economy* (dealing with financial matters). The expanded query for such a topic should include terms from each of these different classes. Otherwise, the expanded query could be dominated by one concept, e.g. recycling, and would then retrieve many non-relevant documents, dealing for example, with the recycling of tin cans.

Even when many of the top-ranked documents are relevant and cover most of the concepts in the query, clustering the top documents and selecting terms from each cluster should be useful, since it would ensure that the expanded query addresses different kinds of relevant documents rather than becoming a one-sided representation of the search topic that would find relevant documents of a particular kind only.

There are some queries, however, for which particular types of non-relevant documents also cluster. For such queries, selecting terms from all clusters would introduce a bias in the query in favor of these non-relevant documents, and performance would deteriorate. Ideally, we would like to be able to select terms from only those clusters that contain useful terms (usually clusters that contain a large proportion of relevant documents). Since we do not know this information *a priori*, we use the following heuristic to select clusters for expansion. Cluster vectors are compared to the original query, and ranked in order of similarity. Terms from the best two clusters are used for query expansion. Hopefully, this method would retain most of the benefits of clustering for queries where it helps, while minimizing the damage for the kind of queries mentioned above where clustering hurts performance.

The actual clustering algorithm used is straightforward. Clusters are initialized with one document per cluster. A *cluster vector* is associated with each cluster. Initially, this vector is simply the vector corresponding to the single document contained in that cluster. The similarities between pairs of clusters are computed using the inner product similarity between the corresponding cluster vectors. The most similar pair of clusters is merged into a single cluster, with the new cluster vector being formed by combining all the document vectors in the two clusters into one large cluster vector. Similarities between clusters are recomputed and the merging process continues until the similarity between the most similar pair of clusters falls below a threshold.

Combining these techniques, our final strategy for the first official TREC 6 run (Cor6A1cls) is the following:

1. Retrieve 1000 documents using the initial query (using *Lnu.ltu* weights).
2. Generate cooccurrence information about the query terms from the top 1000 documents.
3. Rerank the top 50 documents as in TREC 5 (using correlation and proximity information).
4. Assume the top 20 documents relevant, documents ranked 501–1000 non-relevant.
5. Generate clusters for the top 30 documents and save the best (most heavily weighted) terms from each cluster vector.
6. Rank the cluster vectors according to their similarity to the original query (using *bnn* weights for the clusters) and select the best 2 clusters.
7. Expand the query by 100 words and 20 phrases using Rocchio expansion with $\alpha = 8$, $\beta = 8$, and $\gamma = 8$. The expansion terms are selected from among the saved terms for both clusters and the actual number of terms selected from a cluster is proportional to its similarity to the original query.
8. Retrieve the final set of 1000 documents using the expanded query.

SuperConcepts.

It was clear from tuning runs that the cluster approach above was having some effect, but not as much as we hoped query balancing would get. So we developed a last-minute experimental approach, SuperConcepts, that directly attacks the problem of query balance while expanding a query.

The goal of SuperConcepts is to balance the aspects of an original query given any reasonable expansion of that query. In the sample query from above (*What is the economic impact of recycling tires?*), an expansion of it may add 20 terms dealing with *economic* but only 5 terms related to *recycling tires*. Unless we balance the expanded query, we may retrieve only financial documents.

There has been a lot of earlier work dealing with grouping of terms into high-level concepts(e.g., two very different approaches are [14, 6]) but most deal with static definitions of groups. In contrast, SuperConcepts are defined at run-time from query-related terms.

The basic approach is to create SuperConcepts of related terms.

1. Assign each original query term to be the seed of a Superconcept.
2. Assign every expansion term to every correlated SuperConcept seed, dividing its feedback weight proportionally to the correlation.
3. Finally, apportion part of each SuperConcept to other more highly weighted correlated SuperConcepts.

For example, suppose we have an expanded query of (⟨ economic,.6 ⟩ ⟨ recycling,.8 ⟩ ⟨ tires,.9 ⟩ ⟨ financial,.5 ⟩ ⟨ profit,.1 ⟩ ⟨ loss,.1 ⟩ ⟨ Goodyear,.3 ⟩) where the first three terms were the original query terms. Then we might end up with SuperConcepts of

- ⟨ economic,.6 ⟩ ⟨ financial,.5 ⟩ ⟨ loss,.1 ⟩ ⟨ profit,.08 ⟩ ⟨ Goodyear,.05 ⟩
- ⟨ recycling,.8 ⟩ ⟨ profit,.02 ⟩
- ⟨ tires,.9 ⟩ ⟨ Goodyear,.25 ⟩

where *Goodyear* has been allocated mostly to the *tires* SuperConcept, but a bit to the *economic* SuperConcept.

A SuperConcept is matched against a document by matching against the included terms. But rather than including each match as an independent inner product match, we deprecate the importance according to how many other terms of this SuperConcept have matched. More precisely, we sort the SuperConcept terms by decreasing weight, and then match each term in turn. The contribution of a single term match is defined to be

$$1/(1 + cn) * qwt * dwt \tag{2}$$

where c is a constant, n is the number of terms that have previously matched this SuperConcept, and dwt and qwt are the document and query weights of this term.

In the SuperConcept tire example above, if c is 1, then a document matching

- *economic* will have an effective query weight of .6
- *financial* will have an effective query weight of .5
- both *economic* and *financial* will have an effective query weight of .85 (.6 + 1/2 .5).

Thus each additional match of a term related to *economic* will count a bit less, just as multiple matches of the same term normally count as the *log* of the number of matches. Note that $\int 1/(1+n) = \log(n)$.

The final SuperConcept approach is

1. Use a base approach to determine an expanded query
2. Form SuperConcepts from both the original query and the expanded query.
3. Match SuperConcepts against documents, which deprecates multiple terms matching within the same SuperConcept.

Ad-Hoc experiments and analysis

We submitted three runs in the ad-hoc category: Cor6A1cls starts with the description-only queries and uses document clustering during expansion; Cor6A2qtcs clusters the terms in an initial TREC 5 style expanded query into SuperConcepts and uses the SuperConcepts for the final retrieval; Cor6A3cll is identical to Cor6A1cls except that it starts with the full queries instead of the description field only.

Table 1 shows the results for the various runs across 50 queries. The performance level for the short queries is fairly poor in absolute terms. Using full queries improves performance by about 20%. Due to an indexing error, the queries used in our official Cor6A3cll run ontained the description and narrative fields only, but not the title. (NIST inadvertently changed the format of the title field of the topic. Since all of our processing is automatic, we didn't notice.) We reran this run after fixing the problem and obtained a 13% increase over our official average precision figure. This is somewhat surprising in view of the brevity of the title (which usually consists of only 2–3 words), but given that this field contains the essence of the query (in contrast to the description and narrative sections which often contain extraneous terms) this improvement is reasonable.

Run	Average precision	Total rel retrieved	R precision	Precision @100 docs
Cor6A1cls	.1799	2391	.2155	.1794
Cor6A2qtcs	.1809	2332	.2076	.1714
Cor6A3cll	.2139	2590	.2415	.2010
Cor6A3cll (fixed)	.2413	2852	.2650	.2242

Table 1: Ad-Hoc results (50 queries)

Run	Task pool	Best	$\geq$ median
Cor6A1cls	Short automatic	0	37
Cor6A2qtcs	Short automatic	0	37
Cor6A3cll	Long automatic	2	40
Cor6A3cll (fixed)	Long automatic	2	44

Table 2: Comparative automatic ad-hoc results (50 queries)

Table 2 shows that our runs compare reasonably with other runs. For several queries, however, our short-query runs do not work well — the performance of these runs falls below the median on 13 queries. The results for the long queries are somewhat better. The incorrect run is above median on 40 queries and is best on 2. For the fixed run, these numbers are 44 and 2 respectively. (Note that these are only approximate numbers for the fixed run, since the statistics computed from the pool of runs change if the incorrect run is replaced by the correct one.)

We analyze our first run in greater detail in Table 3. The base vector run using single terms only yields a low average precision of 0.1479. Using phrases in the initial retrieval improves performance by almost 8%. Reranking the top 50 documents using cooccurrence and proximity information improves results by another 4%[1]. When the terms in the original vector are reweighted using the assumed relevance information, we get a further improvement of 5%.

Run	Avg. P
1. Vector (*Lnu.ltu*), single terms only	1479
2. above + phrases	1593 (+7.7%)
3. After reranking top 50	1648 (+11.4%)
4. Reweighted vector (no expansion)	1728 (+16.8%)
5. Expansion (no reranking, no clusters)	1831 (+23.8%)
6. Exp (reranking, no clusters)	1804 (+22.0%)
7. Exp (no reranking, clusters)	1825 (+23.4%)
8. Cor6A1cls (reranking, clusters)	1799 (+21.6%)

Table 3: Ad-Hoc component results (50 queries)

The last four rows in Table 3 attempt to analyze the relative importance of the two principal ingredients of our approach — *reranking* to improve the set of 20 documents that are assumed relevant, and *clustering*

[1]Note that, for consistency, we evaluate this run by computing the average precision at 1000 documents. The benefit of reranking the top 50 documents is therefore partly concealed by the unchanged ranks of the remaining 950 documents and is more significant than suggested by the 4% improvement.

to ensure that the final query is a well-balanced one. Unfortunately, the numbers are fairly close to each other and it is difficult to draw reliable conclusions from them. The simplest approach that uses neither reranking nor clustering seems to work well for the TREC 6 task, unlike earlier tasks.

A look at the results for individual queries shows that each technique helps and hurts performance on approximately the same number of queries. For example, comparing the runs corresponding to lines 5 and 6, we find that reranking significantly improves performance (by at least 8%) on 17 queries, but hurts performance on 19. Similarly, comparing the runs corresponding to lines 5 and 7, we find that clustering improves the results for 10 queries but hurts performance for 7. It is clear that if we could predict which techniques would work well on which queries, then overall results could be dramatically improved. However, examination of the queries yields no insights into ways to characterize the queries appropriately. Perhaps we might be able to discern patterns if the number of queries were considerably larger.

We also study the effect of query length on retrieval effectiveness for the TREC 6 task. Table 4 shows the average precision and total number of relevant documents retrieved when various sections of the query are indexed. The base run retrieves 1000 documents per query using just the vector match. The final run starts with the appropriate set of indexed queries but is otherwise identical to Cor6A1cls.

Indexed sections	Title only	Desc only	Title+Desc	Full
Base run	0.1959	0.1593	0.2040	0.2169
	2200	2031	2369	2522
Final run	0.2169	0.1799	0.2306	0.2413
	2483	2391	2680	2852

Table 4: Ad-Hoc results for various query lengths (50 queries)

Traditionally, longer queries have yielded better results. The performance trends shown in Table 4 generally agree with this observation, with one significant exception. The description-only queries do rather badly compared to the extremely short (often single-word) title-only queries. This seems to be because the title section contains the word(s) most crucial to the query, whereas the description is often more verbose and sometimes states minor constraints that have to be satisfied by relevant documents. This introduces extraneous terms which dilute the importance of the core concepts and lead the retrieval process astray.

It also is the case that some queries may have been designed by the assessors with the idea that both the title and the description would be used. The description field very often uses an alternative vocabulary to describe the concept from the title. This makes sense for a user to do if the two fields are to be used together, but hurts if the fields are to be considered separate queries as they were in these experiments.

Routing

Routing Methodology

Continuing the Cornell tradition, we submitted one "conservative" run based on previously developed and well-tested techniques (Cor6R1cc), and one experimental run (Cor6R2qtc) based on recent work.

The Cor6R1cc run uses the same techniques as our TREC 5 routing submission (Cor5R1cc). The development of this approach is presented in detail in [5]. Below, we briefly recapitulate the steps involved in generating the routing queries:

1. Create the initial query vector with *ltu* weights. Inverse document frequency information is obtained from the training collection.

2. For each query, retrieve the top 5000 documents from the training set.

3. Expand the query by adding single terms and phrases that occur in more than 5% and 10% resp. of the relevant documents.

4. Weight the terms in the expanded query using the Rocchio feedback formula. Only those non-relevant documents that are within the query zone are used in this step.

5. Add pairs of cooccurring terms that occur in more than 7% of the relevant documents.
6. Compute weights for the added pairs using Rocchio's formula. Only the top-ranked $2R$ non-relevant documents (where R is the number of relevant documents for the query) are used in this step.
7. Retain the most highly weighted 100 single terms, 10 phrases and 50 cooccurrence pairs in the final query.
8. Run the expanded query through a 3-pass Dynamic Feedback Optimization (DFO) step to fine-tune the weights.

The experimental SuperConcept run, Cor6R2qtc, directly takes the expanded query used in Cor6R1cc, and assigns the expansion terms to SuperConcepts seeded by the original query.

Routing automatic results and analysis

The performance figures for the official Cornell submissions are shown in Table 5. Both the runs are in the automatic category. We recently discovered an error in our indexing script for the test database. The results obtained when this error is corrected (and a few other minor changes made — certain document sections that were being previously omitted are indexed) are shown in the last row. The absolute figures are good, though it is perhaps somewhat surprising that we are unable to do better in spite of the enormous quantity of training data.

Run	Average precision	Total rel retrieved	R precision	Precision @100 docs
Cor6R1cc	.3983	5429	.4198	.3930
Cor6R2qtc	.3766	5233	.3999	.3802
Cor6R1cc (fixed)	.4028	5483	.4174	.3949

Table 5: Automatic routing results (47 queries)

Run	Best	$\geq$ median
Cor6R1cc	3	47
Cor6R2qtc	1	41
Cor6R1cc (fixed)	1	45

Table 6: Comparative automatic routing results (47 queries)

Table 6 compares our submissions to other submissions in this category. The performance of our methods is impressive — Cor6R1cc performs at or above the median on all queries and is best on 3. Cor6R2qtc is only somewhat less consistent with a performance below the median on 6 queries.

The Cor6R2qtc result is disappointing but not too surprising. The DFO optimizes the weights assuming an inner product similarity function. Those weights will not be optimal when distributed across SuperConcepts.

We show the step-by-step analysis of Cor6R1cc in Table 7. The basic vector run starts at an average precision of 0.2640. Simply reweighting the original query terms using Rocchio's formula yields a 14% improvement. Expanding by 100 terms and 10 phrases gives us a considerable improvement of 18%. Adding pairs of cooccurring words improves results marginally, and the final significant improvement in performance comes from optimizing the query term weights. The contribution of each step to the final performance on this task is very similar to that for the TREC 5 task [5]. This is reassuring in terms of the stability of our techniques.

High-Precision

The TREC 6 High-Precision track is a new track this year. It is an attempt to perform a task that is much more closely related to real-world user interactions than the ad-hoc or routing task. The goal is simple: a

Run	Avg. P
1. Vector (*Lnu.ltu*), incl. phrases	2640
2. Reweighted vector (no expansion)	3010 (+14.0%)
3. Expansion by 100 terms, 10 phrases	3485 (+32.0%)
4. Exp by 100 terms, 10 phrases, 50 pairs	3646 (+38.1%)
7. Above + DFO (Cor6R1cc)	3983 (+50.9%)

Table 7: Routing component results (47 queries)

user is asked to find 10 relevant documents in 5 minutes. No other restrictions are put on the user (other than no prior knowledge of the query, and no asking other users for help). Evaluation is simply how many actual relevant documents were found among the 10 documents supplied by the user (Precision at 10 documents).

There are no restrictions on the type of resources the user may use during this task other than

- Only one user per query per run (no human collaboration).
- The user and system can have no previous information about the query (eg, the system cannot have previously built a query dependent data structure.)

In particular, the users are allowed to make multiple retrieval runs, allowed to look at documents, allowed to use whatever visualization tools the system has, and allowed to use system or collection-dependent thesauruses, as long as they stay within the 5 minute clock time.

This track tests (at least) the effectiveness, efficiency, and user interface of the systems. The task provides a forum for testing many of the neat ideas in user interface and visualization that have been suggested over the years.

Unlike other interactive evaluations (for example, the TREC 6 Interactive task), no attempt is made to factor out user differences when comparing across systems. All users are assumed to be experts and equally proficient in use of their own system. This allows for fair comparison of systems, but implies that the absolute level of performance within the track will be better than the level obtainable from casual users. These are upper-bound interactive experiments.

High-Precision Methodology

Our methodology for the TREC 6 high-precision task is very similar to the one we adopted for the TREC 4 Interactive and TREC 5 Manual ad-hoc runs [4, 5]. The user's main task is to provide relevance judgements to be fed to our standard Rocchio relevance feedback algorithm. Direct modification of the query (adding/deleting terms to/from the query) was also occasionally (rarely) used by the searchers. The other principal component of our technique is the use of pipelining or "parallel" processing so that expensive retrieval techniques can be executing while the user continues to make judgements. The details of the method are given below:

1. The current time is noted. The user views the query supplied by NIST and enters it, either as-is or suitably modified, into the system.
2. The query entered by the user is indexed and a set of documents is retrieved using a simple vector match.
3. The top-ranked documents are presented to the user.
4. The user starts viewing the documents and judging them 'relevant', 'non-relevant' or 'possibly relevant'.

   In parallel, a child process is forked to retrieve additional documents using a more sophisticated retrieval algorithm: the initial query is used to retrieve 1000 documents, the top 20 are assumed to be relevant, documents ranked 501-1000 are assumed to be non-relevant, and automatic feedback is used to expand the query by 25 single terms and 5 phrases, using $\alpha = 8, \beta = 8$ and $\gamma = 8$. The expanded query terms are then grouped into superconcepts and this query is run to retrieve the final set of documents. This run corresponds to our official Cor6A2qtcs run.

5. After every judgment, the current time is noted. All documents retrieved so far are sorted such that the documents judged relevant come first, followed by all documents judged possibly relevant, followed by all unjudged documents, and the top 10 documents in this ranking are saved in a file and time-stamped.
6. After every 5 categorical judgments (i.e. 'relevant' or 'non-relevant'), a relevance feedback process is started in parallel if the child process is idle. For this process, documents marked relevant by the searcher are assumed to be relevant, and documents marked non-relevant as well as those retrieved at ranks 501-1000 by the initial user query are assumed to be non-relevant. Documents marked "possibly relevant" are not used in the feedback process. The query is expanded by 25 words and 5 phrases. $\alpha = 8, \beta = 8$ and $\gamma = 8$ are used. While this feedback process is running in the background, the user continues to judge more documents.
7. When the child process is done (i.e. retrieval or feedback completes), and the new retrieval results are available, these results are merged into the current list of top-ranked documents being shown to the user.
8. The final top 10 documents for the query will be the last set of 10 documents saved with a timestamp under the 5-minute limit

User Interface. The user interface for the TREC 6 high-precision runs is a simple textual interface that does not use any windowing[2]. The UI is used to view documents and mark documents 'relevant', 'non-relevant', or 'possibly relevant'. Query term occurrences in document texts can be optionally highlighted. The interface also displays the time elapsed since the beginning of the search. The interface may also be used to modify the query as follows: the text of the current query is shown to the user who makes appropriate changes and submits the modified query, which is then used in all subsequent processing.

Users. Three runs are presented; each the result of one user running all 50 queries. The user and some environmental characteristics are:

1. User 1 - Run HP1
   - Experience: System designer (HP interface designer)
   - Machine: Sparc 20/512 (old low end machine)
   - UI Settings: Highlighting of terms on
2. User 2 - Run HP2
   - Experience: SMART System implementer
   - Machine: UltraSparc 140 (new low end machine)
   - UI Settings: Highlighting of terms off
3. User 3 - Run HP3
   - Experience: System designer
   - Machine: UltraSparc 140 (new low end machine)
   - UI Settings: Highlighting of terms off

[2] It is very similar to the interface used in the TREC 4 interactive task. This interface is described in detail in [4]

All users should be considered experts although User 2 had much less SMART and TREC experience. User 1 was running on a slower machine (by a factor of 2), which undoubtedly had an impact on how many of the more expensive runs finished. Users 2 and 3 decided to turn off highlighting of document terms that occurred in the query. On long documents, highlighting was expensive (2–3 seconds) and it was not felt it was worth it.

The evaluation results are presented in Table 8. The base case is the official run Cor6A3cll which gives the precision at 10 documents of that automatic run. The Cor6HP3 run got close to 2/3 of the optimum performance. It also was greater than or equal to the median on 84% of the queries, being the best (or tied for best) for 36% of the queries.

Run	Precision	Relative Precision	Num queries Best	Num queries $\geq$ Median
Base	.4260	-	-	-
Cor6HP1	.5540	.5564	10	38
Cor6HP2	.5660	.5820	12	39
Cor6HP3	.6020	.6298	18	42

Table 8: High-Precision comparison (50 queries)

One important question is how the users agreed with the official TREC relevance judgements. If the HP track is to have meaning, the disagreement between user interpretation of relevance to a query, and the official assessor interpretation can't dominate the results. Table 9 gives the total number judged relevant, possibly relevant, and non-relevant for each user, along with the corresponding number officially judged relevant. For example, of 690 documents judged non-relevant by User 3, 58 (8%) had been judged relevant by the assessors. The final column shows the number of documents judged non-relevant or possibly relevant for which our trace did not record the docid. For these documents, we cannot tell whether they were officially relevant or not.

In general, from 68% to 77% of the documents judged relevant by the users were judged relevant by the assessors. This agrees with the previous TREC consistency studies done by NIST. The disagreements tended to concentrate on only a few queries. 3/4 of the queries had 0 or 1 disagreements on the user judged relevant documents. But, for example, on query 305 the three users found a total of 24 documents they thought were relevant, with 6 more possibly relevant. Of those 30 documents, none were judged relevant by the assessor. Query 345 had 19 disagreements, and query 309 had 15 disagreements.

Run	User judged Relevant	Official Relevant	User judged Possibly Rel	Official Relevant	User judged NonRel	Official Relevant	Non-relevant Unknown docs
Cor6HP1	300	225	111	40	430	33	188
Cor6HP2	398	270	15	3	527	48	140
Cor6HP3	345	265	103	32	690	58	162

Table 9: High-Precision User-assessor consistency (50 queries)

Examining the figures in Table 8 and Table 9 more closely, there is an apparent correlation between overall effectiveness and number of documents judged. User 1 (on the slow hardware with highlighting) only judged 1029 documents, while User 2 judged 1040 and User 3 judged 1360 documents.

Being on a slower machine caused User 1 problems, in that the more expensive (and hopefully better) iterations did not finish as quickly as for the other users. Even on the faster machine, speed was somewhat a problem. User 3 took that into consideration and used shorter initial queries, averaging 22 words/phrases per query instead of 30 for User 1 and 32 for User 2. The shorter queries shortened the time for the more expensive iterations and thus allowed them to be used for more judgements. Thus User 3 was able to judge 358 documents that were retrieved as the result of feedback (iterations 2-7), while User 1 only examined 24. Table 10 gives the number of retrieved documents per iteration, along with the number judged relevant.

Overall, the high-precision track itself seemed to work. Consistency of judgements did not seem to be much of a problem. The range of evaluation results appears to be good. If the runs were much better (in the 75% precision range, instead of 60%), then the consistency issues may start dominating the comparisons

Run	Rel/Ret Iter 0	Rel/Ret Iter 1	Rel/Ret Iter 2	Rel/Ret Iter 3	Rel/Ret Iter 4	Rel/Ret Iter 5-7
Cor6HP1	236/903	59/102	0/12	5/12	-	-
Cor6HP2	235/719	97/178	52/99	10/33	2/17	2/3
Cor6HP3	165/703	75/299	72/187	25/102	7/48	1/21

Table 10: High-Precision Relevant/Retrieved for each Iteration

between runs. At least for the Cornell runs, the results are very understandable: if you look at more good documents, you get better results!

Cross-lingual

Once again, our emphasis in our multi-lingual runs is to see how effective retrieval can be with a minimal amount of linguistic information. Good linguistic information should someday be able to improve a good non-linguistic (statistical) search. However, prematurely concentrating on the linguistic aspects of operating on different languages may not only yield sub-par retrieval, but indeed may interfere with evaluation of the linguistic approach. A useful linguistic technique tied to a sub-optimal statistical base retrieval may be unfairly judged as not important.

As we did in TREC 3 with Spanish, we spent a small amount of time determining simple stemming rules for French and coming up with a French stopword dictionary. We use these to perform a mono-lingual French-French run, using almost exactly the same technique as for our English SuperConcept run, Cor6A2qtcs. (These cross-lingual runs are described in more detail in later sections.)

We use the same document collection for an English-French cross-lingual run. No dictionaries are used; instead the English query words are treated as potentially mis-spelled French words. The English query is expanded by adding French words from the collection that are lexicographically nearby. This query is then run as a normal mono-lingual run (including more automatic expansion using an initial retrieved set).

Another "cross-lingual" run is English queries against machine translated German documents. Exactly the same techniques as a pure English mono-lingual run are applied.

Finally, we also did a base case English-English run.

French run preparation

The French collection was indexed using a combined French/English stopword list and a French stemmer. The English stopwords were the SMART default list of 500-some; about 300 entries were added for French, starting by going over the 1000 most frequent words from one year of the French collection and adding any pronouns, articles, or common tenses of être and avoir that hadn't already shown up, along with a few translations of English stopwords such as numbers.

While examining the most frequent words from the French collection, it became apparent that a large minority of words that should have had accent marks did not. We decided to drop all accent marks that did exist during indexing to get these different representations of the same word to match each other.

The French stemmer removed *l'*, *d'*, *s'*, *n'*, *qu'*, *j'*, and *m'* from the beginning of words. Final "ment" was removed in hopes of matching adverbs and adjectives. Many verb forms were returned to the infinitive by trimming final "erai", "erons", "erez", "eront", "erais", "erait", "erions", "eriez", and "eraient" to "er", and similarly for the "ir" forms. (Final "era" and "eras" were left alone as occurring too often outside verb forms and not that often as verbs.) A few very rough attempts were made to deal with masculine/feminine and singular/plural, with final "elle", "elles" being trimmed to "el"; "enne", "ennes" to "en"; "if", "ifs" to "iv" (matching trimmed "ive" and more English words); and a fallback removal of final "s", "e", "es", and "'s" (the last due to apparent English contamination in the collection).

Mono-lingual runs

For the French-French mono-lingual run, the French queries are run through the above process before being matched against the French documents. Then the techniques used in our English SuperConcept ad-hoc run, Cor6A2qtcs, are applied to the French collection. The only difference is that no phrase indexing is done. The run does very well when evaluated on the 21 queries for which relevance judgements are available (Table 11). It is greater than or equal to the median run on all but one query, and is best on one query.

Run	Average precision	Total rel retrieved	Num queries Best	Num queries $\geq$ Median
Cor6FFsc	.3815	1078	1	20

Table 11: French-French monolingual run (21 queries)

Our English-English mono-lingual run is a base case for other systems to compare against. This is exactly the Cor6A2qtcs ad-hoc run except with the cross-lingual English queries and documents. One thing to note about the results in Table 12 is the absolute level of performance. The average precision is more than twice as high as the ad-hoc run. This should help allay some fears about the low level of TREC 6 ad-hoc performance; when put in a comparatively easy environment (easy queries and smaller collection), the absolute level of performance is good.

Run	Average precision	Total rel retrieved	Num queries Best	Num queries $\geq$ Median
Cor6EEsc	.4202	1159	7	19

Table 12: English-English monolingual run (21 queries)

English-French runs

The French/English cross-language retrieval was based on the idea that the languages share many, many cognates. English swallowed much of French vocabulary (although not grammar) quite recently as language developments go. Lists of the most frequent non-stopword terms from English and French news collections suggested that 10-15% of terms would, modulo stemming, match exactly across languages, while another 25-30% were close enough that there was a reasonable hope that they could be matched automatically.

A trie was built of all indexed terms from the French collection occurring at least five times, in the hopes that we could massage English cognates to match these French terms. Terms extracted from the English queries, using the standard English stemmer, were run against the trie, and French terms found were added to the query. Besides the general procedures of adding or deleting one letter, two equivalence classes of letters were defined after studying the frequent English and French terms. One equivalence class was vowels, such that an English term would pick up French terms with the same consonant pattern but different vowel sequences. The other was k sounds, where any combination of c-k-qu could substitute for any other. Naturally, some unwanted terms were added at this stage, but so were a large number of cognates.

The next step to improve retrieval was to automatically expand the query by adding terms occurring in the top 20 ranked documents (just as we do in pure English retrieval). We assume that correct cognates are sufficient to ensure the top documents are in the general subject area of the query. The expansion terms from the top documents should give us related pure French non-cognates. The new terms were weighted according to our normal Rocchio formulation, and the new query was then run against the collection again to give us our final results. The SuperConcept approach used in the mono-lingual run was not used here, though in future runs it could be.

In one of the runs, a small thesaurus of about 300 words (half time and geographic references, half common or important words that were not cognate between the languages) was used to automatically add additional French terms to the query. This thesaurus was prepared from general knowledge before the queries were known. Using the thesaurus increased recall in several cases, but had no particular effect on precision.

This very simple, non-linguistic approach did very well! As Table 13 shows, half of the queries work reasonably just using cognates (11 of 21 are at or above the median). When the very small thesaurus/dictionary is added, another 5 queries perform above the median.

Run	Average precision	Total rel retrieved	Num queries Best	Num queries ≥ Median
Cor6EFent	.1760	541	2	11
Cor6EFexp	.1982	689	3	16

Table 13: English-French cross-lingual run (21 queries)

Query-by-query results comparing the mono-lingual run against the cross-lingual run were about as would be expected from the discussion above. For queries #1 (Waldheim/Nazi/crime) and #14 (international terrorism(e)), the terms came out the same in both languages. For query #6, the terms were similar in both languages, but the English query happened to use "air" while the French query used "atmospherè", causing the English query to do better because more documents talked about air pollution than atmospheric pollution. For queries #5 (medicine vs. médecine) and #10 (solar vs. solair(e)), vowel substitution worked nicely. For query #9, main terms like "deforestation" matched, but others like "flood"/"inondation" did not, depressing results from those of the French query. For query #19, "wine" and "vin" did not match, although that would be within the abilities of a more robust cognate-finder. For queries #17 (potato vs. pomme de terre) and #24 (teddy bear vs. ours en peluche), there was no hope for an automatic match without a major dictionary, although "teddy bear" did occur twice in the French documents.

In this initial investigation, we used a human to come up with the rules for what extra types of "misspellings" should be considered (i.e., vowel substitution and c-k-qu equivalence). However, there is no reason why these rules cannot be learned automatically for any pair of related languages. Just trying all transformations of words in one language, and seeing which transformations often end up with a word in the other language should work. This can be explored in the future.

The approach used here is only useful for related languages. We need to discover how related the languages need to be for reasonable performance. There are large numbers of former colonies in Africa, the Caribean, and elsewhere, whose everyday language has drifted from that of the former colonial power. This result suggests that we do not need to consider those as separate languages, with distinct linguistic support needed for retrieval.

English-TranslatedGerman runs

Our English-TranslatedGerman run is just a quick run to see what happens if we treat it entirely as a mono-lingual English run. We did no analysis of any factors that might make the translated environment different from normal English, and just ran our standard English ad-hoc SuperConcept run. Table 14 shows that we are above the median for 11 out of the 13 queries, though the absolute level of performance does not seem very exciting.

Run	Average precision	Total rel retrieved	Num queries Best	Num queries ≥ Median
Cor6ETGsc	.1858	393	1	11

Table 14: English-TranslatedGerman cross-lingual run (13 queries)

Chinese

Last year we participated in the first Chinese track and had one of the very top results. It was a very low effort track for us; nobody from our group understood any Chinese at all and we had no training data, so all we could do was run our basic approach treating single Chinese characters as words, and pairs of Chinese characters as phrases.

This year we still have no one who understands any Chinese, but we have last year's work as training data, so the effort involved has been a bit greater.

Unfortunately, most of our simple attempts to improve last year's statistical results showed very little improvement. The final official submissions are based on this year's English ad-hoc run using SuperConcepts, Cor6A2qtcs. Official Chinese run Cor6CH1sc follows exactly the same procedure as the English run, except it was decided to treat the two-character phrases as being the base concepts of the SuperConcepts instead of the single terms as in the English run.

The second official run, Cor6CH2ns, is exactly the same as Cor6CH1sc, except no expansion single characters were added. Instead of adding 15 single terms and 15 phrases to the original query from the top 20 initially retrieved documents, only 25 phrases were added.

Table 15 shows the results. There is disappointingly little difference between the two runs. Both runs did very well, with the Cor6CH2ns being greater than or equal to the median on 18 out of the 26 queries.

The absolute performance level achieved by everybody is extremely impressive. The median performance level (averaging the median average precision for all queries) of .535 is by far the highest level of performance of any TREC track in history. However, it remains unclear why this level of performance is being achieved. Is it a property of the Chinese language, perhaps due to less ambiguity of important terms? Or is it a property of these particular Chinese queries, prepared by assessors who know the vocabulary used in the target document set very well?

Run	Average precision	Total rel retrieved	Num queries Best	Num queries $\geq$ Median
Cor6CH1sc	.5547	2765	0	16
Cor6CH2ns	.5552	2763	0	18

Table 15: Chinese automatic ad-hoc (26 queries)

Comparison with past TREC's

It is difficult to determine how much systems are improving from TREC to TREC since the queries and the documents are changing. For example, in TREC 3 the "Concept" field of the queries was removed. These terms proved to be very good terms for retrieval effectiveness in TREC 1 and TREC 2; thus the TREC 3 task without them is a harder task than previous TRECs. The TREC 4 task was more difficult since so much more of the text was removed from the queries. TREC 5 and TREC 6 continued using short queries which seem more difficult. Also, the average number of relevant documents per query has been steadily reduced every year, going from 328 in TREC 1 to 92 in TREC 6. Very broad (and easy) queries have been eliminated.

To examine both how much SMART has improved over the years of TREC, and how much harder the TREC ad-hoc tasks have gotten, we ran our 5 TREC SMART systems against each of the 5 TREC ad-hoc tasks. Table 16 gives the results. Note that the indexing of the collections has changed slightly over the years so results may not be exactly what got reported in previous years. In the interest of speed, we ran our current implementation of the query and document indexing and weighting. Things have changed more than we expected; a number of the figures from the earlier runs on the earlier collections are off by as much as 3–5%, probably due to the indexing changes. The results, though, are all consistent with each other.

Comparing the columns of Table 16 gives an indication of how much harder the TREC task has gotten during the 5 years of TREC. Five quite different versions of the same system all do from 45% to 65% worse, in absolute numbers, on the TREC 6 task as compared to the TREC 1 task. The TREC 1 and TREC 2 figures are about the same. Performance starts to drop in TREC 3 and 4 when the queries get progressively shorter. The short high-level queries of TREC 5 and TREC 6 prove very difficult for all versions of SMART.

Comparing the rows of Table 16, it is obvious that our results with our TREC 6 approach are not noticeably different from our TREC 5 approach. We were disappointed; we had gotten some improvements on the smaller collections we had been tuning with, but they didn't carry over to any of the full TREC tasks. This is more evidence of the standard maxim that you must test approaches on as many collections and environments as possible. This is especially true of the more modern approaches that expand and change the query drastically. These approaches dependent on features of the individual queries and it is easy to get

Methodology and Run	TREC 1 Task	TREC 2 Task	TREC 3 Task	TREC 4 Task	TREC 5 Task	TREC 6 TI+DESC
TREC 1: ntc.ntc	.2442	.2615	.2099	.1533	.1048	.1224
TREC 2: lnc.ltc	.3056	.3344	.2828	.1762	.1111	.1472
TREC 3: lnc.ltc-Exp	.3400	.3512	.3219	.2124	.1287	.1499
TREC 4: Lnu.ltu-Exp	.3628	.3718	.3812	.2773	.1842	.2006
TREC 5: Exp-rerank	.3759	.3832	.3992	.3127	.2046	.2278
TREC 6: Rrk-clust	.3765	.3835	.4011	.3073	.1978	.2356
% Change from ntc.ntc	+54	+47	+91	+100	+89	+92

Table 16: Comparisons of past SMART approaches with present

off track for several queries (and this is what we see when we analyze results on a query-by-query basis). 50 queries on one document collection are not enough to be convincing anymore.

Conclusion

The Cornell SMART Project is again a very active participant in the TREC program with these TREC 6 runs. With the exception of the high-precision relevance feedback runs, everything we have presented here is completely automatic and uses no outside knowledge base (other than a small list of stopwords to ignore while indexing). Manual aids to the user can be built on top of this system to provide even greater effectiveness.

We use two approaches, clustering and SuperConcepts, to try to preserve query balance while expanding a query for the ad-hoc task. Both have been mildly successful on other TREC collections (3% – 7% improvement) but help only marginally, if at all, on the main TREC ad-hoc task (Description field only queries). One explanation for this could be that the Description field is not intended to be a stand-alone query, but intended to be used in conjunction with the title field. (The very short title field perform significantly better by itself than the longer description field.)

For routing, we unsuccessfully tried to use SuperConcepts to balance the query. It is obvious more work needs to be done there. We also submitted an official run using our TREC 5 routing approach; that run is still one of the best runs, doing better than the median for all queries.

Our entries in the new High-Precision track are based almost entirely on relevance feedback. Our users read documents and judge them, rather than attempting to manually reformulate a query, or use query visualization techniques to focus in on relevant documents. Results are much better than the median.

We participated in the Chinese track, but little new work was done. Results are better than the median for 2/3 of the queries; a nice result considering nobody in our group understands a word of Chinese.

Our French-French mono-lingual run is very successful, above the median on all queries. The only language related work done for this run was the construction of simple stemming rules and a simple stopword list.

Our English-French cross-lingual run is surprisingly successful. We use almost no linguistic information, and just treat the English query terms as mis-spelled French words that need to be corrected. This suggests that retrieval across related languages does not need a lot of apparatus to be effective.

For all the tasks, though particularly for the cross-lingual, it is very clear there is a lot of variation between queries. Some methods work well for some queries but poorly for others. In order to understand these effects, we need to perform our experiments on larger numbers of queries. Accurately characterizing queries is likely to be the next big area of improvement for automatic information retrieval.

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, James Allan, and Amit Singhal. Automatic query expansion using SMART : TREC 3. In D. K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*. NIST Special Publication 500-225, 1995.

[4] Chris Buckley, Amit Singhal, and Mandar Mitra. New retrieval approaches using SMART : TREC 4. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*. NIST Special Publication 500-236, 1996.

[5] Chris Buckley, Amit Singhal, and Mandar Mitra. Using query zoning and correlation within SMART : TREC 5. In D. K. Harman, editor, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. NIST Special Publication 500-???, 1997.

[6] Scott Deerwester, Susan Dumais, Tom Landauer, G. Furnas, and R. Harshman. Indexing by latent sematic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] 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.

[8] 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.

[9] M.A. Hearst. Improving Full-Text Precision on Short Queries using Simple Constraints. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, April 1996.

[10] Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO '97*, 1997.

[11] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214. Association for Computing Machinery, 1998.

[12] S Robertson, S Walker, S Jones, M M Hancock-Beaulieu, and M Gatford. Okapi at TREC 3. In D. K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*. NIST Special Publication 500-225, 1995.

[13] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In Hans-Peter Frei, Donna Harman, Peter Schauble, and Ross Wilkinson, editors, *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. Association for Computing Machinery, 1996.

[14] Padmini Srinivasan. Intelligent information retrieval using rough set approximations. *Information Processing and Management*, 25(4):346–361, 1989.