behind that figure for ad-hoc this year. Maybe next year...!

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 and Gerard Salton. Optimization of relevance feedback weights. In Ed Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357, New York, July 1995. ACM.

[3] 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.

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

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

Methodology and Run	TREC 1 Task	TREC 2 Task	TREC 3 Task	TREC 4 Task	TREC 5 Task	% Change from TREC 1 task
TREC 1: ntc.ntc	.2431	.2594	.2095	.1507	.1038	-57
TREC 2: lnc.ltc	.3078	.3352	.2801	.1626	.1072	-65
TREC 3: lnc.ltc-Exp	.3474	.3534	.3209	.2022	.1254	-63
TREC 4: Lnu.ltu-Exp	.3609	.3762	.3760	.2841	.1833	-49
TREC 5: Exp-rerank	.3765	.3830	.3972	.3146	.2065	-45
% Change from ntc.ntc	+55	+48	+90	+109	+98	

Table 16: Comparisons of past SMART approaches with present

vector should be. That leaves little room for new term expansion on the very long queries of TREC 1 and TREC 2. The results here suggest we need to go back to our earlier expansion strategy for long queries.

The lnc.ltc weighting strategy becomes less effective with the short queries of TREC 4 and TREC 5 (as compared to ntc.ntc). One reason is simply the tf component of the query weight is useless with the short queries. But in addition to that, there are indications that idf is more important in short queries than long. This needs to be studied in the future.

I am still somewhat surprised that we and other groups can continue to improve our systems as consistently as we have been able to. It really is impressive!

Conclusion

The Cornell SMART Project is again a very active participant in this year's TREC program. With the exception of our one relevance feedback manual run, 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.

Most of our effort this year was spent on the routing task. Overall, our routing performance improves by 23% over last year's approach. We re-examined our parameters and introduce the idea of "query zones" which contain documents which may be related to the domain of the query, but may not be relevant. Terms useful in distinguishing the query zone documents from the rest of the collection may not be useful in distinguishing relevant from non-relevant documents within the collection.

We also add dynamic pairs to our expansions. These are a high precision device intended to separate the relevant documents from the top non-relevant documents. Pairs of terms that co-occur in the relevant documents are chosen to be added to the query.

Once again, DFO (Dynamic Feedback Optimization) substantially improves routing retrieval performance. The combination of the added dynamic pairs and DFO worked very well (suggesting that possibly our initial weights on the dynamic pairs could be substantially improved in the future).

Our ad-hoc expansion approach improves 12% over last year's very impressive results. The major source of the improvement is reranking the top 50 initially retrieved documents using a High-Precision approach to end up with 20 documents that individually cover the concepts of the query. These documents prove to be more useful as a source for expansion and reweighting information.

Our manual results show that minimal involvement by users, just having users judge the relevance of documents, can result in a very effective retrieval set. Any user can judge whether documents are useful; that is their purpose in coming to the information retrieval system in the first place.

The Spanish results are again considerably above the median, and used no language knowledge. The runs were exactly the same runs as we run on the English tasks.

The Chinese results are also extremely good and are done with no knowledge at all of Chinese or of Chinese word segmentation. This may suggest that segmentation is a minor issue for retrieving Chinese and shouldn't be a major focus.

We had been improving our ad-hoc results by 20-25% a year. We concentrated on routing and fell a bit

Run	Average Precision	Total Rel Retrieved	R Precision	Precision 100 docs
Cor5C1vt (vector)	.3266	1286	.3598	.3026
Cor5C2ex (expansion)	.3598	1343	.3829	.3084

Table 14: Chinese Ad-hoc - 19 queries

Run	Average Precision
Vector, single terms only, Lnu.ltu	0.2859
Vector plus phrases (Cor5C1vt)	0.3266 (+14%)
Vector + phrases + reweighting	0.3063 (+7%)
Expansion, 501-1000, top50 reranked (Cor5c2ex)	0.3598 (+26%)

Table 15: Chinese components results - 19 Queries

Tables 14 and 15 give more details of the evaluations of the runs and the components going into the runs. The expanded query does 10% better than the basic vector version. That is much less than the improvement on the English short query run. However, starting with a long English query, expansion only improves the English result by 14% (.2234 for a base vector with phrases as compared to .2544 for Cor5M1le). We only ran the long versions of the Chinese queries, so the difference in percentage improvement between Chinese and English is not all that great.

One interesting result that bears further investigation is that expansionless reweighting of the original query terms and phrases based on the top retrieved documents actually hurts performance. The third run of Table 15 is 5% worse than the second run. This compares to a 5–10% typical improvement in English and Spanish. One conjecture is that very common single characters start dominating the phrases, but this remains un-examined.

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. As was discussed earlier in this paper (see Table 7) the TREC 5 task is even more harder than TREC 4, apparently because of the types of queries.

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%. We'll investigate these and may have a revised table at the conference. The results, though, are all consistent with each other.

The last column 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 5 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 got progressively shorter. The short high-level queries of TREC 5 proved very difficult for all versions of SMART.

A couple of points of interest about our results in Table 16. Our TREC 5 approach does not get as much improvement on the TREC 1 and TREC 2 tasks as it does on later tasks. The TREC 5 expansion, instead of adding a specified number of terms and phrases as in previous years, specifies what the final length of the

Base vector single terms only	Base vector single terms and phrases	Rocchio re-weighting no expansion	Rocchio expansion
3154	3091 (-2.0%)	3170 (+0.5%)	3949 (+25.2%)

Table 12: Analysis of Cor5SP1s

Run	Best	$\geq$ median	$<$ median
Cor5C1vt	1	16	3
Cor5C2ex	3	19	0

Table 13: Comparative Chinese Results (19 queries)

Spanish Discussion

Phrases are constructed purely statistically — any pair of adjacent non-stopwords that occur sufficiently frequently in the corpus are deemed to constitute a phrase. Thus, the method used to generate phrases for our English collection was used to generate phrases for the Spanish collection as well. The use of phrases did not improve retrieval, however. For the base vector run using inner product similarity with the initial queries, the average precision dropped from 0.3154 to 0.3091 when phrases were used. Similarly, expansion by 25 single terms and 5 phrases (our official submission, which gave an average precision of 0.3949) is no better than expansion by 25 single terms only (average precision 0.3951).

A similar observation holds for this year's English runs as well. The use of phrases in the simple vector match run for Cor5A1se yields an improvement of only 1.5% compared to about 5% in previous years, and the use of phrases during automatic expansion improves performance by only 1.4% as against about 12% in previous years. We need to investigate why the usefulness of phrases seems to have diminished.

Accented characters occur inconsistently in the AFP corpus i.e. the same word occurs both with and without an accent on some letter. We need to investigate ways to handle these inconsistencies.

Chinese

If the linguistic effort Cornell spent on Spanish was small, the effort spent on Chinese was minuscule. There were no people involved at Cornell who understand anything about the Chinese language. We knew that Chinese characters occupied two bytes, and we discovered that punctuation all began with one of two byte values. Other than that, we can do no linguistic analysis and therefore can do no segmentation or stopwords. Every Chinese character (except punctuation) is retained and treated as a separate word within SMART.

The changes to SMART for chinese involved about 30 lines of code in the tokenizer, and two lines of code in the phraser (phrase components are not separated by whitespace in Chinese). Other than that, no changes were made to SMART or our basic retrieval algorithms. The most human time spent on the Chinese track was spent finding a Chinese text pager on the Web, and getting it to work.

The Chinese collection is indexed in the same fashion as English. A list of Chinese "phrases" are automatically formed by taking all adjacent pairs of non-punctuation characters that occur more than 20 times in the test collection. This list is deliberately made larger than our English list of phrases (174,891 entries as opposed to 158099 entries) since we know that no Chinese word segmentation will be used.

Other than the larger list of phrases, we index Chinese exactly like English, and our retrieval runs corresponded exactly to the runs we would do in English, including weighting and ad-hoc expansion. The Cor5C2ex run repeats the Cor5A1se exactly (including determining the top 20 documents by reranking the top 50 with a High-Precision filter) except we expand by 15 single terms and 15 phrases instead of 25 single terms and 5 phrases. The other official run, Cor5C1vt, is a straight simple Lnu.ltu weighted vector run.

Our non-linguistic approach does remarkably well. Table 13 shows that the Cor5C2ex run is at or above the median for all 19 queries, having the best results on 3 queries. Even the simple vector match is only below the median on 3 queries.

to achieve with relevance feedback. In practice, the improvement should be even more since these test result values are lowered due to the well-known mismatch between the test user's notion of relevance in the on-line judgements, and the assessor's notion of relevance for evaluation. We have not yet studied the mismatches between the two, but plan to in the future.

Spanish

The multi-lingual track demonstrates the language-independent nature of SMART. The techniques used in this year's ad-hoc task were directly applied to the Spanish task. The modifications needed to enable retrieval in Spanish were done for TREC 3 [4] and took a total human time of 5 – 6 hours.

Cor5SP1s and Cor5SP2l

Cor5SP1s and Cor5SP2l, the two runs submitted by Cornell, use identical techniques. The only difference is that Cor5SP1s indexes only the Spanish description field (<S-desc>) of the query, while Cor5SP2l indexes the Spanish narrative field (<S-narr>) of the query as well. The queries were weighted with *ltu* weights, and the documents with *Lnu* weights. 1000 documents were retrieved using simple inner product similarity with the initial query. The 50 top-ranked documents were re-ranked as in the ad-hoc run Cor5A1se. The top 20 documents in the resulting ranking were assumed to be relevant and documents ranked 501 – 1000 were assumed to be non-relevant. Rocchio relevance feedback was then used to expand and reweight the query (25 single terms and 5 phrases were added). The 1000 documents retrieved using this query constitute our official submission.

Spanish Results

The results for the two runs are shown in Table 10 and Table 11. Both runs did well compared to the median score of the groups participating in the Spanish task. As expected, the absolute figures are better when long queries are used.

Run	Best	$\geq$ median	$<$ median
Cor5SP1s	1	18	7
Cor5SP2l	3	18	7

Table 10: Comparative Spanish Results - 25 Queries

Table 12 shows the average precision obtained at each stage of the Cor5SP1s run. A simple vector match using single terms only yields an average precision of 0.3154. Our official result (0.3949) is 25.2% above this baseline. The improvement is mostly due to the addition of new terms to the query from the top-ranked documents. When the query is not expanded, but simply re-weighted using Rocchio feedback under the assumption that the 20 top-ranked documents are relevant, the results do not improve appreciably[1].

For Cor5A1se, the English equivalent of Cor5SP1s, the improvements obtained are compatible — the average precision for the submitted run (0.2065) is 39.2% above a lower baseline of 0.1484 corresponding to the base vector run.

[1] The improvement due to re-weighting is greater (5.3%) for the long queries (Cor5SP2l).

Run	Average Precision	Total Rel Retrieved	R Precision	Precision 100 documents
Cor5SP1s	.3949	2103	.3890	.3448
Cor5SP2l	.4323	2222	.4468	.3796

Table 11: Spanish Ad-hoc - 25 queries

Run	Average Precision
Vector, single terms only, Lnu.ltu	0.1484
Vector plus phrases	0.1506 (+1%)
Vector + reweighting	0.1619 (+9%)
Expansion, no non-rel docs, no reranking	0.1856 (+25%)
Expansion, 501-100 non-rel docs, no reranking	0.1909 (+29%)
Expansion, no non-rel docs, top50 reranked	0.2070 (+39%)
Expansion, 501-1000, top50 reranked (Cor5A1se)	0.2065 (+39%)
Expansion, 501-1000, top50, Corr Pairs (Cor5A2cr)	0.2109 (+42%)
TREC 4 Ad-hoc on TREC 5 task	.1833

Table 8: Ad-hoc components results - 50 Short Queries

Run	Average Precision	Total Rel Retrieved	R Precision	Precision 100 docs
Cor5M1le (automatic)	.2544	3092	.2845	.2272
Cor5M2rf (manual)	.2931	3085	.3112	.2412

Table 9: Comparison of manual vs. automatic Ad-hoc 50 queries

The two major new attempts at improvement this year are using a query zone to define the non-relevant documents to use within the Rocchio formula, and reranking a close query zone of 50 documents to find the best 20 to use (basically a High-Precision filter). Using documents ranked 501-1000 gives a marginal improvement by itself, but this improvement disappears when the High-Precision top 20 documents are used. We do not know why this happens, but the improvement of using the 501-1000 non-relevant documents by itself was small enough so it may not be worth investigating.

The High-Precision reranked 20 document set gave a substantial improvement of 12% over the top 20 document set. We were very pleased with those results, especially since our investigations into High-Precision retrieval are just beginning. This offers great opportunities for improvement in the future.

Adding all the machinery to calculate co-occurrence pairs and weight with correlations did not yield any improvement at all. Cor5A2cr is better than Cor5A1se on 30 out of the 50 queries, but the differences were marginal. This needs to be explored further, but it is possible that better relevance information is needed.

Manual Ad-hoc relevance feedback run

For the first time, Cornell has submitted a run in the Manual Ad-hoc category. This is a repeat of the run we did last year in the Interactive track. Two expert users (authors of this paper) were given 25 queries each, and the initial retrieval rankings of the automatic run Cor5m1le, and told to judge relevance of documents for 5 minutes per query. On average, each user looked at about 25 documents per query, though a large number of those documents were easily dismissed. These relevance judgements were then used within the Rocchio framework to construct a new feedback query, expanding by 25 terms and 5 phrases, but with no co-occurring pairs. This new query was then automatically run from scratch (no frozen ranks of the user judged documents), and the top 1000 documents submitted to NIST. No other manual effort was made, and the user judgements were used only to construct the new query.

The only manual input was relevance judgements, which any user should be able to make accurately. Thus the expertise of the users should not be a factor here, other than a complete system novice might take longer because of unfamiliarity with the user interface.

The results in Table 9 show we got a 15% improvement overall. The relevance feedback improved performance on 37 out of the 50 queries. There were two queries in which the users could find no relevant documents.

The 15% improvement is very substantial, showing the effectiveness increase that any user should be able

Run	Task Pool	Best	$\geq$ median	$<$ median
Cor5A1se	Short Automatic	3	40	10
Cor5A2cr	Short Automatic	2	41	9
Cor5M1le	"Manual"	2	30	20
Cor5M1le	Long Automatic	7	42	8

Table 5: Comparative Automatic Ad-hoc Results (50 queries)

Run	Average Precision	Total Rel Retrieved	R Precision	Precision 100 docs
Cor5A1se (short)	.2065	2849	.2331	.2002
Cor5A2cr (short)	.2109	2848	.2404	.2018
Cor5M1le (long)	.2544	3092	.2845	.2272

Table 6: Ad-hoc results - 50 queries

Further evidence of that comes from the low values of the absolute levels of retrieval effectiveness. If queries are harder, then the top 20 retrieved documents will contain fewer relevant documents. Thus it will be easier for the query expansion to sidetrack the final query into areas not related to relevance.

Table 6 shows the absolute levels of performance are very low; in fact lower than they have ever been in TREC. (TREC 1 official figures were worse, but only because they used a different evaluation methodology). The level of performance is due to the queries getting substantially harder rather than the systems deteriorating. A subjective look at the queries suggests that many queries this year are hard because they are much more high-level than normal. Later we present a table showing how each of the TREC 1-5 Cornell systems perform on each of the TREC 1-5 ad-hoc tasks. It clearly shows the queries getting harder every year, while the systems improve.

Another measure of the toughness of the various TREC ad-hoc tasks is seeing how well the best system for each query does. If no system does well on a query, then it can be called a hard query. You would expect these "best system" numbers to improve substantially each year as the systems improve, and as the number of participating systems increase. But as can be seen from Table 7, the numbers have remained constant for several years before taking a sudden dip this year. So systems have improved at about the rate that the queries have gotten harder until this year when the queries became much harder and outstripped the system improvements!

Table 8 looks at the components giving the performance on the ad-hoc task. The single term vector performance starts off at a low base-line value. Adding phrases helps only marginally, probably because the queries are all very short and not many phrases occur in them. Assuming the top 20 documents are relevant and reweighting the original query terms gives a substantial 9% improvement, as we can better tell what terms of the original query are extraneous and can be de-emphasized.

Another major improvement comes when terms from the top 20 documents are added to the query and weighted with Rocchio. This run corresponds to our TREC 4 expansion run, but with some minor tweaks. The TREC 4 result is given here for comparison sake, and indeed is almost identical.

TREC task	Average over all queries of Best run at 100 / MIN (NumRel, 100) (lower numbers imply poorer performance)
TREC 1	0.414
TREC 2	0.653
TREC 3	0.676
TREC 4	0.672
TREC 5	0.556

Table 7: TREC task hardness measure

248 (*What are some developments in electronic technology being applied to and resulting in advances for the blind?*), we find that the terms *advances* and *technology* are strongly related, whereas the terms *advances* and *blind* are not well related. Then a new query zone consisting of the top 50 documents initially retrieved is defined.

For each document in this tight query zone, the document is reindexed and broken up into 50 term, overlapping window, chunks. Each window is compared to the original query. The matching terms are considered in order of rarest first. Each matching term contributes its idf value times a factor inversely proportional to the maximum correlation the term has to all previous terms that have been considered in this window. I.e., if a term is highly correlated (among the top 1000 documents) with a previously considered term, then the match is not important and the idf value is deprecated. In the above example, if we have seen the term *advances* in a document, we would not consider the presence of the term *technology* to be a strong new match; but we will consider the presence of the term *blind* to be a strong new match, just because it is not well related to the previously seen terms.

The 20 documents with the highest matching windows are determined using the above algorithm. These documents are the High Precision result set, and will be assumed to be relevant in the next stage.

Analogously to our routing task approach, we want to define a non-relevant set of documents that are in the same domain as the query, though still non-relevant. The initial query zone of 1000 documents can be used for this; we make the assumption that documents ranked 500 through 1000 are non-relevant. For some queries with lots of relevant documents this may be a bad assumption, but it will often be reasonable.

The normal Rocchio expansion described in the routing section is then performed, except that the original query must be emphasized more, since the relevance information is much less dependable. Thus the original query weight counts as much as weights in the relevant and non-relevant documents: the $A.B.C$ Rocchio parameters are set to values 8.8.8. Note that in previous years we used 8.8.0 and ignored the non-relevant documents altogether. We add 25 terms and 5 phrases to the original query, choosing those terms and phrases with highest Rocchio weight. Terms which occurred in fewer than 4 "relevant" documents and phrases which occurred in fewer than 2 "relevant" documents were ignored.

Our official automatic run Cor5A1se uses this as the final query. The other official automatic run, Cor5A2cr, continues adapting our full correlation routing approach to the ad-hoc task. Co-occurrence pairs are defined as in the routing task. They are weighted by a combination of the Rocchio weights of the terms, and the correlation of the terms within the top 1000 documents, with no dependence on relevance (again, the relevance information is much less dependable). The top 15 weighted co-occurrence pairs are added to query.

Ad-hoc automatic results and analysis

Our runs do comparatively well on the ad-hoc task, though not as well as they did last year. This year it is tougher to judge comparative performance from the median figures given all the fine-grained distinctions between runs. We complicated matters by "sneaking" an automatic run in under the manual category. As in past TRECs, we try to present one automatic run (Cor5A1se) that has been well developed over the past year, and one experimental run that gives a new untested approach that may or may not work (Cor5A2cr as described above). These two short query runs took up our 2 automatic slots, so we put our long query automatic run (Cor5m1le) in a manual slot, where it did quite respectably against the real manual queries. Cor5m1le is exactly the same run as Cor5A1se, except it starts with the full version of the query instead of the Description field only.

Table 5 compares our three runs against the published medians of their respective task pools. Cor5M1le is compared against both the Manual pool, where it is officially located, and against the Long Automatic pool where it belongs. It would be best on 7 queries within the Long Automatic pool, and at or above the median on about 42 queries. (We say about 42 since the addition of Cor5M1le to the pool will change the medians by some unknown amount.)

These runs are all strongly above average, but they are not as consistent across queries as we would have hoped. This suggests our expansion techniques may have gotten off-target and expanded inappropriately.

Run	Average Precision
Vector Lnu.ltu - idf of learning set	.2462
Vector Lnu.ltu - idf of test set	.2612
Ad-hoc expansion of test set	.2877

Table 4: Learning Set vs Test set Routing Runs - 45 Queries

old learning set data. We can't answer the question fully, but we ran the test set of documents as a separate full ad-hoc collection to get some ideas. The first two runs of Table 4 only differ in that the first run uses idf values determined from the learning set of documents, and the second run uses idf values determined from the test set of documents. There's a 6% difference between the two; certainly enough to have a noticeable effect, but probably not enough to pose serious problems to the testing methodology.

We were also interested in just how much more the real learning set judgements help in expansion as opposed to the "fake" judgements used in our ad-hoc expansions that arbitrarily assume the top 20 to 30 retrieved documents are relevant. The third run of Table 4 gives the results of running our TREC 5 ad-hoc run (described below in the ad-hoc section) on the ad-hoc version of the test collection. As expected, the improvement over the basic vector run is much smaller. The .2877 figure can be compared against the .3365 result of the real relevance feedback without DFO (which obviously can't be done in the ad-hoc case). The difference is 17%, certainly significant but somewhat disappointing considering the massive amount of relevance information available. This suggests that there is still room for improvement in our routing approaches; we should be able to do better.

Ad-Hoc Methodology

In TREC 4 we established a very good basic vector weighting model ("Lnu.ltu") that we continue to use unchanged for TREC 5. Our TREC 5 efforts have been directed at improving our query expansion procedures. The overall expansion approach we use is to perform an initial retrieval upon the collection, retrieving a small number of documents. Those documents are assumed to be relevant and are submitted to a relevance feedback expansion and weighting stage similar to that described above in the routing section.

There are two steps in the above procedure on which we can improve. The first is to improve the initial retrieval so that the small number of documents assumed to be relevant are in fact more likely to be relevant. The second is to improve the expansion procedure once those relevant documents are obtained.

Mandar Mitra at Cornell has been working on the first step. The basic goal is to produce a "High-Precision" retrieved set: a set using retrieval techniques that may throw away lots of relevant documents in an effort to be sure that the remaining documents are likely to be relevant. This trading of recall for precision can be approached in many ways. The basic approach we use is "Query Coverage" (QC), parts of which has been worked on under various names by a number of groups like Xerox.

In "Query Coverage", a query zone around the query is defined. These documents are candidates to be retrieved. Each document is submitted to an evaluation measure that attempts to determine how many query concepts are present within small windows of the document. If numerous unrelated query terms are discovered within a document window, then the multiple key concepts of the query are likely to be covered, and the document is likely to be relevant.

Note that many relevant documents may not satisfy the QC criteria. But since the goal here is precision, not recall, ignoring those documents will not hurt as long as other relevant documents are found.

The simplest sort of QC algorithm is to count the number of different matching terms between query and document window. But this has the problem that phrases and strongly related terms can dominate. A three-word phrase match would be judged as important as if three distinct concepts matched. This happens often, so something more complicated is needed.

For TREC 5, our QC algorithm initially retrieves 1000 documents in the query zone. The correlations between all the original query terms terms are calculated within this zone. For example, for TREC query

Run	Best	$\geq$ median	$<$ median
Cor5R1cc	7	44	1
Cor5R2cr	5	42	3

Table 1: Comparative Routing Results (45 queries)

Run	Average Precision	Total Rel Retrieved	R Precision	Precision 100 docs
Cor5R1cc	.3842(.3807)	4249	.4137	.3296
Cor5R2cr	.3433(.3759)	4313	.3693	.3411

Table 2: Routing results - 45 (39) Queries

There are obviously a large number of alternatives throughout this entire process. During the algorithm development we have only partially explored the options and expect we will alter the process in the future. This is especially true of the correlation weighting, which was added a few days before the routing deadline. There is much work to be done here!

Routing experiments and analysis

Cornell submitted two official routing runs, both automatic and both using the above process. Cor5R1cc uses co-occurrence of terms for pairs with the Rocchio weighting. As a more experimental run (i.e., we had no idea whether it would work) Cor5R2cr weights pairs with the correlation measure described above.

Table 1 shows that both runs performed very well and very consistently. Cor5R1cc was less than the median on only one query! Table 2 gives the absolute scores. The figures in parenthesis give the Average Precision for the 39 queries with more than 2 relevant documents. Cor5R2cr did poorly on two of those queries that Cor5R1cc did well on.

A head to head comparison shows the results were more different from each other than might be expected from the similarity of the averages. Each beat the other by more than 10% on exactly 10 queries.

If we break out some of the various components of the routing runs, we can get some insight as to what is important. The basic vector run in Table 3 starts at an average precision of .2462. Adding plain Rocchio reweighting of the original query terms improves results by 13%. Then there's a big jump of 19% when terms and phrases are added to go up to 100 terms and 10 phrases, reweighted using Rocchio. Adding co-occurrence pairs gains marginally, but then DFO on top of that is another 19% gain, with some large part of that being the reweighting of the co-occurrence pairs. (We don't yet know how to accurately weight those pairs.) All-in-all we end up with the routing judgements giving us a 56% improvement, which is much greater than in previous years. As can be seen from the last line in Table 3, our TREC 5 routing approach is a very substantial 23% improvement over the TREC 4 approach run on the same TREC 5 data.

One question that often comes up when talking about TREC routing is just how much of a problem is the mismatch between the learning set of documents and the test set of documents. Due to the difficulty of obtaining test collections, it is rare that the new TREC routing data can be considered a continuation of the

Run	Average Precision
Vector Lnu.ltu (including phrases)	.2462
Vector + reweighting	.2818 (+13%)
Vector + reweighting + exp	.3242 (+32%)
Vector + reweighting + exp + cooc:	.3365 (+37%)
Above, plus DFO (Cor5R1cc)	.3842 (+56%)
TREC 4 Routing on TREC 5 task	.3133

Table 3: Routing components results - 45 Queries

but never with great success. The object is to add pairs of terms that co-occur comparatively more often in the relevant documents than the non-relevant. Since the addition of pair information is primarily a high precision device, we decided to focus on a very narrow query zone. We consider co-occurrences among the relevant documents plus the top 2R non-relevant documents (where R is the number of relevant documents). Candidate pairs are those pairs of terms with at least one term occurring in the original query, and the other term being either a query term or an expansion term.

The weight of the candidate pair is given by the Rocchio formula, with the weight in a relevant or non-relevant document defined to be the minimum of the weights of the two component terms in that document. (Document weights at this point in the processing are defined without idf; idf is added later, and for pairs is the actual idf of the pair in the collection.) Like the single terms and phrases, pairs to be considered must meet a minimum occurrence threshold. For TREC 5 this was set to 7%; i.e., all candidate pairs must occur in at least 7% of the relevant documents.

Another option for defining the weight of a candidate pair is looked at in the second of our TREC 5 official runs. Instead of using the Rocchio formula, the pair weight is defined by the correlations of the pair and involved terms to relevance.

$$\begin{aligned} \text{PairWt} &= \text{Correlation\_of\_Pair\_to\_relevance} \\ &* (\text{Correlation\_of\_}T_1\text{\_to\_}T_2\text{\_in\_relevant\_docs} \\ &\quad - \text{Correlation\_of\_}T_1\text{\_to\_}T_2\text{\_innreldocs}) \end{aligned}$$

An ideal pair would be one in which not only is the pair highly correlated with relevance, but the terms of the pair are more likely to co-occur together in the relevant documents than the non-relevant. The latter factor will emphasize strong independent contributions of the two terms to relevance. (If the latter factor is low, but the first factor is high, then the pair is probably a phrase and already included in the phrase component.)

Unlike in previous years, the choice of the final added single terms, phrases, or pairs, is deferred until after Rocchio weighting is done. At that time, the pairs with the highest Rocchio weights are kept (here, 100 single terms, 10 phrases, and 50 pairs). We also do not guarantee that original query terms will be kept in the final query. Our weighting schemes seem much more robust than in the past, so all this is feasible.

After the Rocchio (and possibly correlation) weighting is done and the query concepts have been selected, the weights are further optimized using a three-pass DFO algorithm. Using more passes as in TREC 4 does not seem to change performance, but is considerably more time consuming.

A summary of the steps used in routing, given a learning set, L, of judgements:

1. Create the initial vector query with ltu weighting from L.
2. Find the top 5000 documents to the query in L.
3. Expand query with single terms and phrases occurring in more than 5% or 10% of relevant documents
4. Weight expanded query using non-relevant documents from the query zone within the Rocchio formula.
5. Add pairs of co-occurring terms, one of which must be an original query term, occurring in more than 7% of relevant documents.
6. Weight pairs either according to Rocchio weights or correlation weights.
7. Restrict expanded query to 100 terms, 10 phrases, and 50 pairs, using those concepts with highest weights.
8. Perform a 3-pass DFO to fine-tune the weights.

- Explore the concept of a "query zone". The properties of non-relevant documents somehow related to the query (not having low similarity) are different from those of the general non-relevant documents.
- Re-examine co-occurrence of pairs of terms to see if pairs commonly occurring in the relevant documents can improve retrieval.
- Explicitly use the correlation of pairs of terms to weight pairs added by a co-occurrence examination.

In TREC 4, we started with a basic Rocchio feedback approach, which modified the indexed weights of query terms based on the presence of terms in the judged relevant and non-relevant documents.

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

The Rocchio parameters A,B,C had values 64,64,2 in TREC 4. We expanded by 50 single terms and 10 phrases that occurred in the relevant documents. We then modified these Rocchio weights by undergoing a six pass Dynamic Feedback Optimization (DFO) stage[2].

The original query weight was heavily emphasized in TREC 4, while the weight among the non-relevant documents was almost negligible. However, given the increased accuracy of our new weighting schemes, we can rely much more on the weights in the documents. For TREC 5, we use parameters 8,64,64 which are dominated by the weights in the documents as opposed to the original query. We also double the number of single terms being added to 100. The number of phrases is kept at 10, though additional pairs are added as described below. A final tweak is to only allow terms to be added if they occur in a certain percentage of the relevant documents. This prevents the randomly occurring terms from entering the query. For TREC 5 this randomness-threshold is set to 10% for single terms and 5% for phrases.

In a vector space model of information retrieval such as that used by SMART, a "query zone" can be thought of as a volume of the vector space which surrounds the query. It can be a tight query zone, consisting of those document vectors strongly related to the query vector, or a weak zone, composed of documents that have some undetermined relationship with the query. In either case, the properties of those documents in a "query zone" will be different from the properties of the vast majority of the documents in a large collection.

One use of a query zone is to define a set of documents that are hopefully within the same domain of the query, but not relevant to the query. The basic original query can easily distinguish a target set in which most relevant documents will appear; but the efforts to do so may interfere with the efforts to distinguish relevant from non-relevant documents within the target set. For example, the term "computer" may be a very good term for distinguishing the domain for a query about 'which disk drive should I get for my Mac', but it is not going to be useful within the technical domain. If the term is weighted too heavily (e.g., if it occurs in nearly all the relevant documents, but only 5% of the much larger collection of non-relevant documents), general articles about computers may get ranked too highly.

For TREC 5, we use a query zone of 5000 documents to calculate the Rocchio weights (actually we use 5000 non-relevant documents, plus the judged relevant documents). This doesn't affect the weights due to occurrences in the relevant documents, but the average weight of terms in the non-relevant documents is changed dramatically. Terms that serve only to identify the domain of the query will get substantially downweighted by this process.

This process can also have a positive effect. A term can be a relatively common term in the entire collection but can be a good term for distinguishing relevant documents from the non-relevant documents in the query zone. An example would be a term like *tire* for the query *recycling of tires*. Such a term will get a higher Rocchio weight when weights are learned in the query zone in place of the entire collection.

In the routing environment this implies the system either keeps around the top 5000 indexed documents retrieved for each query, or does a new retrieval upon a retrospective collection for each invocation of the user routing profile. Either is somewhat expensive, but neither option is prohibitive.

The next area for exploration in routing is co-occurrence of terms. This has been explored by Cornell and other groups on and off for at least 20 years, most notably with the probabilistic dependency models,

Singhal for TREC 4. 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 ([3, 1, 4, 5]) are used for TREC 5. 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.

System Description

The Cornell TREC experiments use the SMART Information Retrieval System, Version 12, 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 12 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 12 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 5 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.

Routing Methodology

For the past two TRECs we have concentrated on the ad-hoc tasks. In TREC 3 we worked on expansion of ad-hoc queries using techniques from relevance feedback. In TREC 4 we re-examined basic weighting approaches, particularly looking at document length normalization as a source of improvement.

For TREC 5 this year, we shift back to a focus on the routing task, with the hope that lessons learned there will help improve our results on the ad-hoc task. Our changes centered in four areas, each of which will be discussed:

- Adjust our routing parameters to the more accurate weights developed in the TREC 4 ad-hoc task.

Using Query Zoning and Correlation Within SMART : TREC 5

Chris Buckley*, Amit Singhal, Mandar Mitra

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 5, performing runs in the routing, ad-hoc, and foreign language environments. The major focus this year is on "zoning" different parts of an initial retrieval ranking, and treating each type of query zone differently as processing continues. We also experiment with dynamic phrasing, seeing which words co-occur with original query words in documents judged relevant. 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.

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.

The basic "tf*idf" weighting schemes used within SMART have been discussed many times. For TREC 5 we use the same basic weights and document length normalization as were developed at Cornell by Amit

*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.