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

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

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

[15] Gerard Salton and Chris Buckley. Global text matching for information retrieval. *Science*, 253:1012–1015, August 1991.

[16] Gerard Salton, Chris Buckley, and James Allan. Automatic structuring of text files. *Electronic Publishing*, 5(1):1–17, March 1992.

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

		X.Y	*A.B.C*	R-prec	Total Rel	recall-prec
1.	no fdbk	0.0	8.0.0	3382	6509	2869
2.	no expand	0.0	8.8.4	3531	6849	3087
3.	little expand	20.10	8.8.4	3756	7192	3345
4.	`crnlR1`	300.50	8.16.4	4273	7764	3952
5.	`crnlC1`	varies	varies	4367	7808	4091

Table 6: Routing evaluation

For TREC 2, this failure is ameliorated by the base level performance of the global run. If the correct weights are used, the effectiveness of automatic indexing is extremely good.

Automatic massive query expansion proves to be very effective for routing. Conventional relevance feedback techniques are used to weight the expanded queries. Parameters for the relevance feedback algorithms are estimated both over all the queries and for each query individually. The individual query estimation perform better (3–4%) but by an insufficient amount to be convincing.

References

[1] 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.

[2] James P. Callan and W. Bruce Croft. An evaluation of query processing strategies using the tipster collection. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–355, June 1993.

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

[4] Norbert Fuhr and Chris Buckley. Automatic structuring of text files. *ACM Transactions on Information Systems*, 9(3):223–248, 1991.

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

[6] D. Harman. Towards interactive query expansion. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval*, pages 321–331. Association for Computing Machinery, 1988.

[7] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–68, June 1993.

[8] E. Ide. New experiments in relevance feedback. In Gerard Salton, editor, *The SMART Retrieval System—Experiments in Automatic Document Processing*, chapter 16. Prentice Hall, Englewood Cliffs, NJ, 1971.

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

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

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

Query	Phrase Import	Single Expand	Phrase Expand	Orig Query	Weight Relevant	Weight Non-relevant
51	0.0	500	·	8	24	4
52	0.0	200	·	8	16	0
53	1.0	500	50	8	48	4
54	0.0	200	·	8	36	4
55	0.0	200	·	8	24	8
56	1.0	0	0	8	8	4
57	0.0	500	·	8	24	4
58	1.0	500	0	8	48	4
59	0.5	100	10	8	8	4
60	1.0	0	0	8	8	4
61	0.0	0	·	8	8	4
62	0.0	500	·	8	24	4
63	1.0	0	0	8	8	4
64	0.0	500	·	8	16	4
65	1.0	200	0	8	16	4
66	1.0	200	0	8	16	4
67	0.0	500	·	8	16	4
68	1.0	300	0	8	48	4
69	1.0	500	0	8	24	4
70	1.0	500	50	8	48	4
71	0.0	500	·	8	8	4
72	1.0	300	0	8	48	4
73	0.0	500	·	8	16	4
74	1.0	100	100	8	8	4
75	0.0	100	·	8	24	6
76	1.0	300	0	8	24	4
77	0.0	500	·	8	8	4
78	0.0	0	·	8	24	4
79	0.0	500	·	8	24	4
80	1.0	500	0	8	24	4
81	1.0	500	0	8	32	4
82	1.0	500	0	8	32	4
83	0.0	500	·	8	24	4
84	0.0	100	·	8	36	4
85	1.0	100	10	8	24	4
86	0.0	30	·	8	24	4
87	1.0	500	0	8	48	4
88	1.0	500	50	8	48	4
89	1.0	500	0	8	48	4
90	0.0	500	·	8	16	4
91	1.0	0	0	8	24	4
92	0.0	100	·	8	36	4
93	1.0	100	50	8	8	4
94	0.0	200	·	8	36	4
95	0.0	200	·	8	16	8
96	1.0	300	50	8	48	4
97	0.0	200	·	8	36	4
98	0.0	500	·	8	24	4
99	0.0	50	·	8	24	4
100	1.0	500	50	8	48	4

Table 5: Optimum routing parameters, query-by-query

X	number of single terms to add (possible values 0 to 500)
Y	number of phrases to add (0 to 100)
A	relative importance of original query (fixed at 8)
B	relative importance of average weight in relevant documents (4 to 48)
C	relative importance of average weight in non-relevant documents (0 to 16)
P	relative importance of phrases in final retrieval as compared to single terms (0, 0.5, or 1.0)

Table 4: Parameters of routing

Just re-weighting the query terms according to Rocchio's algorithm gives a 7% improvement. Adding a few terms (20 single terms + 10 phrases) gives 17% improvement over the base case, and expanded by 350 (300+50) terms results in a 38% improvement.

The official run `crnlC1` is actually a bit disappointing. It only results in a 3% improvement over the `crnlR1` run, which is not very significant considering the effort required. Few people are going to keep track of 158 test runs on a per query basis. It may be practical to keep track of 4 or so main query variants, but then the improvement would probably be less than 3%. We are conducting experiments in this area currently.

An open question is the effectiveness of varying the feedback approach itself between queries. Preliminary experiments using Fuhr's RPI ([3]) weighting schemes in addition to the Rocchio variants show larger improvements. In general, RPI (and the other probabilistic models) perform noticeably better than Rocchio if there is very little query expansion, though quite a bit worse under massive expansion. We expect that the combination of RPI for those queries with little expansion and Rocchio for other queries will work well.

One benefit of the `crnlC1` run not entirely represented by the evaluation figures is that retrieval performance is more even. Potential mismatches between feedback method and query are far less likely. `crnlC1` does reasonably on all the queries (above the median system for every query when compared against the other systems).

Routing Implementation and Timing

The original routing queries are automatically indexed from the query text, and weighted using the "ltc" weighting scheme (equation (1)). Collection frequency information used for the idf factors is gathered from D12 documents only. Relevance information about potential query terms is gathered and stored on a per query basis. For each query, statistics (including relevant and non-relevant frequency and total "ltc" weights) are kept about the 1000 most frequently occurring terms in the D12 relevant documents. For TREC 2, this is done by a batch run taking about 90 CPU minutes. In practice, this would be done incrementally as each document was compared to the query and judged. The statistics amounted to about 40,000 bytes per query.

Using these statistics, and the decided upon parameters for the feedback process (A, B, etc.), actual construction of the final query takes about 0.5 seconds per query.

Retrieval times vary tremendously with length of query. We ran in batch mode, constructing an inverted file for the entire D3 document set ("lnc" document weights) and then comparing a query against that inverted file. Not only is this not what would be done in practice, but it is much less efficient than would be done in practice given our massive expansion of queries: for each query in `crnlR1`, well over half the entire inverted file was read! CPU time per query ranged from about 5 seconds (no expansion) to 65 seconds (expansion by 500 terms).

Conclusion

No firm conclusions can be reached regarding the usefulness of combining local and global similarities in the TREC environment. In some limited circumstances minor improvements can be obtained, but in general we have not (yet!) been able to take advantage of the local information we know should be useful.

relevant documents, and the non-relevant documents (A,B,C above); and then exactly what terms are to be considered part of the final vector.

In TREC 1, a similar approach originally proposed by Ide was used.[8, 13] That seemed to work well for the fragmentary relevance information available for TREC 1. TREC 2 has considerably more information available—and of much higher quality—so Rocchio's approach is more appropriate.

The best parameter values to use for Rocchio's algorithm were investigated by splitting the TREC 1 collection into two parts along the natural D1 (TREC 1 initial collection) and D2 (TREC 1 routing collection) lines. D1 formed the learning set and D2 the evaluation set for a large number of experimental runs determining these parameters. The original TREC 1 routing queries (Q2) are expanded and weighted using Rocchio's algorithm with the relevance information from D1. They are then evaluated by running them against D2 and using the known Q2–D2 relevance information.

Queries are expanded by adding the "best" X single terms and the "best" Y phrases to the original query. We used a simple notion of "best" for TREC 2: terms that occurred in the most relevant documents (ties were broken by considering the highest average weight in the relevant documents).

There is a core set of 158 runs using different parameter values for both expansion and weighting. Table 4 gives the six parameter possibilities The trends noticeable in this investigatory set of runs are:

1. Overall effectiveness increases strongly as the number of terms added increases, up until 200 terms at which point it starts to level off.

2. Phrases are reasonably important (6% difference) at low single term expansion numbers, but become less important at higher values (1% difference)

3. As expected, weights in relevant documents are far more important than weights in non-relevant documents.

The parameters of our official run, `crnlR1` are: adding $X = 300$ single terms, adding $Y = 50$ phrases, importance of original query of $A = 8$, importance of weight in relevant documents of $B = 16$, importance of weight in non-relevant documents of $C = 4$, and relative importance of phrases at retrieval time of $P = 0.5$.

Query-by-Query Parameter Estimation

We examined the results for the 158 test routing runs in more detail, query by query. For each of the 50 queries, we found the best test run. The results (see Table 5) show some interesting patterns not brought out by the overall averages. Not surprisingly, the parameters used for `crnlR1` are not best for any single query; they are just a reasonable compromise. There seem to be two main groups of queries: one in which very limited expansion is useful (even 6 queries where *no* expansion is preferred); and one in which the more terms are added, the better (23 queries with expansion of 500 single terms). If massive expansion is useful, in general the original query is less important than the expanded terms: A is much less than B. There is another separate distinction between those queries where phrases are useful and those where phrases appear useless: 1 query worked best adding 100 phrases, 6 with 50 added, 2 with 10, 16 using the original phrases only, and 25 using no phrases at all.

If we retrospectively choose the best parameters for each query (something that cannot be done in practice), then we achieve roughly a 10% improvement. This is substantial enough to actually try a predictive run, so our second official run (`crnlC1`) uses query-by-query choice of parameter values in a predictive (as opposed to retrospective) fashion. The values given in Table 5 were used.

Routing Results

Both `crnlR1` and `crnlC1` do extremely well in comparison with other TREC 2 routing runs:

Run	Best	$\geq$ median	$<$ median
`crnlR1`	7	40	3
`crnlC1`	5	45	0

Evaluation measures in Table for both the official and some non-official runs show the importance of query expansion. Run 1 is the base case original query only (ltc weights).

Name	Num	Description
Pairs	I	Query sentences vs. doc sentences
	II	Query paragraphs vs. doc paragraphs
	III	Entire query vs. doc paragraphs
Which	a	Top matching pair
	b	Non-zero matching pairs
	c	Pairs where similarity exceeds threshold
	d	*All* pairs
Value	1	Similarity (avg)
	2	Number of common terms (avg)
	3	Top matching term (avg)
	4	Count of pairs

Table 3: Local values considered for LSP weighting (all combinations, choosing one from each category)

Future work

We are currently investigating the use of regression analysis to find correlation between relevance and local similarity values. Using such analysis will allow the local values to be selected for cause rather than solely because of experience and intuition. If successful, it will also provide a collection-independent method of selecting which local values are useful. Note that this approach does require a training set of queries and relevance judgements.

We are interested in applying these techniques to the TREC collections with a more useful definition of "paragraph." [17, 2] suggest the possibility of narrowing the search window to fixed-size pieces, ignoring paragraph boundaries. Hearst's "TextTiling" approach ([7]) is intriguing for the topic-coherent units of text it produces.

Routing

In this work, routing queries are formed in two distinct phases. In the first phase, concepts which occur often in relevant documents are added to the original query to expand the vocabulary used. In the second phase, the original concepts *plus* the added concepts are weighted based upon their occurrences in relevant and non-relevant documents.

In TREC 1, query expansion was a major obstacle. It was clear that only very limited expansion was useful, and indeed the best automatic routing run ([5]) used no expansion at all. Thus the original plans for TREC 2 routing included extensive investigation into very selectively adding concepts to queries.

However, as work on TREC 2 progressed it became obvious that the TREC 1 results were somewhat anomalous. For the routing approaches used in this work, selectivity of added terms is not an issue. Rather, the more terms that are added, the better the result—up to a point of diminishing returns. This result agrees with our experiences on the (small) feedback test collections that we have worked with in the past. The original TREC 1 training data for routing was extremely sketchy and the resulting unusual query expansion results were probably due to the lack of information about what a representative relevant document looked like.

The basic routing approach chosen is the feedback approach of Rocchio ([9, 13]). Expressed in vector space terms, the final query vector is the initial query vector moved toward the centroid of the relevant documents, and away from the centroid of the non-relevant documents.

$$\begin{aligned} Q_{\text{new}} &= A * Q_{\text{old}} \\ &+ B * \text{average_wt_in_rel_docs} \\ &- C * \text{average_wt_nonrel_docs} \end{aligned}$$

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

The parameters of Rocchio's method are the relative importance of the original query, the

Run	R-prec	Total Rel	recall-prec
`crnlV2`	3640	8018	3163
`crnlV2-b`,	4053	8256	3512
`crnlV2-b`, (no *not's*	4061	8254	3560
`crnlL2`	3641	8224	3258
`crnlL2-b`	3922	8379	3538
sentence restricted	3960	8252	3477

Table 2: Ad-hoc results

of the query against the paragraphs of the candidate document (III.a.1 from Table 3), with the query terms weighted 1 iff present, and the document terms weighted using formula 1 above (that used by the query in the global similarity).

We then use the global/local values in a series of retrieval runs using the same queries but against the entire TREC 1 document set (D12). We tried a range of α and β values and use the best values for the official run, `crnlL2`. The formula used for `crnlL2` is:

$$\text{sim} = 100 \cdot \text{global} + 16 \cdot \text{local}$$

where "global" is the query/document similarity described above ("ltc-lnc"), and "local" is the top query/paragraph similarity.

It takes roughly 5 hours clock time to determine the suggested weighting coefficients, though multiple combinations of values could be weighted simultaneously—in one case, we calculated each of the 48 possible local variables simultaneously. Each of the retrospective runs takes from 60 to 90 minutes to run, depending on its complexity. These runs take an unusually large amount of time (compared to `crnlV2`) since they require re-indexing from scratch a large number of documents. The basic procedure is to discover the top 1750 documents for each query using the global similarity. Then each of those documents is re-indexed, breaking it down into its component parts (e.g., paragraphs). Then each component part is compared against the query to obtain local similarities.

Other Experiments

The Smart indexing procedures that are used in our experiments do not analyze the documents or queries for negative terms such as *not*. A query which explicitly requests documents "*not* about the United Kingdom or Canada" will actually match any document with those terms. Removing the negative keywords results in insignificant improvement: 16 queries are helped, 16 are hurt, all in only a minor fashion. These results suggest that other terms in the query were more important for locating the relevant documents.

Earlier experiments with an on-line encyclopedia ([14, 16]) demonstrated that precision can be improved by discarding documents which fail a local context check (cf. [1] where such documents were merely given lower similarity measures). That approach on the TREC 2 queries and collection yields almost exactly the same performance as `crnlV2-b` (see "sentence restricted" in Table 2). [1] discusses probable reasons for the limited success of this method.

Analysis of Ad-hoc Results

The results of Table 2 suggest that there is little advantage to using local values in combination with global matches. From run `crnlV2` to run `crnlL2` there is negligible improvement, `crnlL2` does retrieve an additional 120-200 relevant documents.

The retrospective runs using the Wall Street Journal sub-collection suggested there would be greater improvement between `crnlV2` and `crnlL2` than actually occurred. The most obvious problem is that the definition of a paragraph is sub-collection dependent. Our results were tailored to the WSJ sub-collection and probably did not apply well to the other sub-collections where "paragraphs" might be extremely large.

Run	Best	$\geq$ median	$<$ median
`crnlV2`	1	38	11
`crnlL2`	4	40	6
`crnlV2-b`	9	36	5

Table 1: Comparative Ad-hoc results

Global Similarity

The `crnlV2` run demonstrates the quality of results obtainable with simple methods. The weighting for terms is chosen based upon results from TREC 1. Query terms are weighted by the formula in equation (1) ("ltc" in Smart's vocabulary). Document terms are weighted using a normalized logarithmic term frequency ("lnc"):

$$d_{ik} = \frac{\log f_{ik} + 1.0}{\sqrt{\sum_{j=1}^{t} (\log f_{ij} + 1.0)^2}} \qquad (3)$$

where d_{ik} is the weight of term T_k in document D_i, f_{ik} is the occurrence frequency of term T_k in document D_i, and t is the total number of terms in the collection. The denominator provides normalization of vector length. Note the absence of the "idf" factor $\log(N/n_k)$.

Table 2 shows the results of that weighting scheme in `crnlV2-b`. Regrettably, because of an oversight during the official run (a misnamed inverted file), the official submitted run `crnlV2` did not use the weighting approach described above (and recommended in our TREC 1 report). Instead `crnlV2` accidentally used the "idf" factor in both the query *and* the document. That mistake caused 10% loss in retrieval effectiveness (from a recall-precision average of 0.3512 to 0.3163).

Global and Local

Cornell's TREC 1 ad-hoc submission increased the similarity measure of a query and document if some sentence in the query matched some sentence in the document sufficiently well.[1] The result was that any query/document pair which contained a sentence match was retrieved before all that did not have such a match. For TREC 2, we hoped to find a less restrictive balance between the global and local similarities. At the same time, we wished to investigate local similarities using parts other than sentences, and to investigate combining *multiple* local similarities.

Our approach is similar to that used in [4]. We built a training collection using the 50 queries from Q2 and the 74,520 documents from the Wall Street Journal included in D2. For each of the 3.7 million query/document pairs, we calculate the global similarity and some set of local similarity values. The least squares polynomials (LSP) approach developed for [4] are used to find the "ideal" coefficients for the global and local values in the equation:

$$\text{sim} = \alpha \cdot \text{global} + \beta_1 \cdot \text{local}_1 + \beta_2 \cdot \text{local}_2 + \ldots$$

(The LSP functions actually yield a constant factor which we ignore since it does not affect ranking.)

We consider local values from the following broad classes:

- Comparing sentences of the query against sentences of the document. In general, we use a simple "tf×idf" weight without normalization, though we experimented with other weights.
- Comparing paragraphs of the query against paragraphs of the document. (For the most part, each section of the query topic is a separate paragraph.) In this case, we use the weighting of equation 1 above for the query paragraphs, and try a variety of weights for the document paragraphs.
- Comparing the query against paragraphs of the document. We use the same weighting schemes as above.

We also tried combinations of the above categories: e.g., the best matching paragraph pair *and* the best matching sentence pair. See Table 3 for a complete list of local values that were considered.

In all, we tried 72 combinations of local and global values, using from one local value to 19 different local values.[1] The LSP-determined α and β_i's of those values are then applied to a retrieval run on that same set of queries and documents. The top performing result includes only a single local value: the best match

[1] There were roughly 1.2 million possible combinations; we chose 72 that seemed, based on earlier experiments, likely to succeed.

term vector as follows:

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

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

Information retrieval and text linking systems based on the use of global text similarity measures such as that of expression (2) will be successful when the common terms in the two vectors are in fact used in semantically similar ways. In many cases it may happen that highly-weighted terms that contribute substantially to the text similarity are semantically distinct. For example, a sound may be an audible phenomenon, or a body of water.

TREC 1 ([1]) demonstrated that local contexts could be used to disambiguate word senses, for example rejecting documents about "industrial salts" when given a query about the "SALT peace treaty". Overall, however, the improvement in effectiveness due to local matching was minimal in TREC 1. One reason for this is the richness of the TREC queries. Global text matching is almost invariably sufficient for disambiguation. Another reason is the homogeneity of the queries. They deal primarily with two subjects: finance, and science and technology. Within a single subject area vocabulary is more standardized and ambiguity is therefore minimized.

One other potential reason for the unexpectedly slight improvement is that most of the information from local matches is simply being thrown away. Local matches are used as a filter to reject documents that do not satisfy a local criteria: the overall global similarity used for ranking is changed only by the addition of a constant indicating the local match criteria was satisfied. The positive information that a long document might have a single paragraph which very closely matched the query is ignored.

For TREC 2, we look at combining global and local similarities into a single final similarity to be used for ranking purposes.

The other focus of our TREC 2 work is taking advantage of the vast quantity of relevance judgements available for the routing experiments. In TREC 1, the relevance information was fragmentary and even occasionally incorrect. It was hard to use this information in a reasonable fashion. Happily, the results of the TREC 1 experiments furnished a large number of very good relevance judgements to be used for TREC 2. Conventional vector-space feedback methods of query expansion and reweighting are tuned for the TREC environment in the routing portion of TREC 2.

System Description

The Cornell TREC experiments use the SMART Information Retrieval System, Version 11, and are run on a dedicated Sun Sparc 2 with 64 Mbytes of memory and 5 Gbytes of local disk.

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

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

Ad-hoc Results

Cornell submitted two runs in the ad-hoc category. The first, `crnlV2`, is a very simple vector comparison. The second, `crnlL2`, makes use of simplified least squares analysis and a training set to combine global similarity and partwise similarities in a meaningful ratio. Both systems performed at or above the median in almost all queries, as can be seen in in Table 1. The `crnlV2-b` run is the same as the official `crnlV2` run, but with an error in the experimental procedure corrected (discussed below).

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

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

A well-known term weighting system following that prescription assigns weight w_{ik} to term T_k in query Q_i in proportion to the frequency of occurrence of the term in Q_i, and in inverse proportion to 'the number of documents to which the term is assigned.[12, 10] Such a weighting system is known as a tf×idf (term frequency times inverse document frequency) weighting system. In practice the query lengths, and hence the number of nonzero term weights assigned to a query, varies widely. To allow a meaningful final retrieval similarity, it is convenient to use a length normalization factor as part of the term weighting formula. A high-quality term weighting formula for w_{ik}, the weight of term T_k in query Q_i is

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

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

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

The terms T_k included in a given vector can in principle represent any entities assigned to a document for content identification. In the Smart context, such terms are derived by a text transformation of the following kind:[10]

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

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

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

Phrases

The same phrase strategy (and phrases) used in TREC 1 ([1]) is used for TREC 2. Any pair of adjacent non-stopwords are regarded as potential phrases. 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 (D1, TREC 1 initial collection). Phrase weighting is again a hybrid scheme where phrases are weighted with the same scheme as single terms, except that normalization of the entire vector is done by dividing by the length of the single term subvector only. In this way, the similarity contribution of the single terms is independent of the quantity or quality of the phrases.

Text Similarity Computation

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

Automatic Routing and Ad-hoc Retrieval Using SMART : TREC 2

Chris Buckley*, James Allan, and Gerard Salton

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 the TREC 2 environment, performing both routing and ad-hoc experiments. The ad-hoc work extends our investigations into combining global similarities, giving an overall indication of how a document matches a query, with local similarities identifying a smaller part of the document which matches the query. The performance of the ad-hoc runs is good, but it is clear we are not yet taking full advantage of the available local information.

Our routing experiments use conventional relevance feedback approaches to routing, but with a much greater degree of query expansion than was done in TREC 1. The length of a query vector is increased by a factor of 5 to 10 by adding terms found in previously seen relevant documents. This approach improves effectiveness by 30–40% over the original query.

Introduction

For over 30 years, the Smart project at Cornell University has been interested in the analysis, search, and retrieval of heterogeneous text databases, where the vocabulary is allowed to vary widely, and the subject matter is unrestricted. Such databases may include newspaper articles, newswire dispatches, textbooks, dictionaries, encyclopedias, manuals, magazine articles, and so on. The usual text analysis and text indexing approaches that are based on the use of thesauruses and other vocabulary control devices are difficult to apply in unrestricted text environments, because the word meanings are not stable in such circumstances and the interpretation varies depending on context. The applicability of more complex text analysis systems that are based on the construction of knowledge bases covering the detailed structure of particular subject areas, together with inference rules designed to derive relationships between the relevant concepts, is even more questionable in such cases. Complete theories of knowledge representation do not exist, and it is unclear what concepts, concept relationships, and inference rules may be needed to understand particular texts.[11]

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

Automatic Indexing

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

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

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 actu-

*Department of Computer Science, Cornell University, Ithaca, NY 14853-7501. This study was supported in part by the National Science Foundation under grant IRI 89-15847.