CSIRO Routing and Ad-Hoc Experiments at TREC6

Arkadi Kosmynin

Research Data Network Co-operative Research Centre
& CSIRO Mathematical and Information Sciences
723 Swanston St, Carlton, Victoria 3053, Australia

Arkadi.Kosmynin@vic.cmis.csiro.au

Background

CSIRO stands for Commonwealth Scientific and Industrial Research Organization. It is the Australian Government's main research body. This is the first year CSIRO is taking part in TREC. We got involved in textual information retrieval research as a part of our activities in Resource Discovery Unit at the Research Data Network Co-operative Research Centre. The primary aim of our research in IR is improvement of the efficiency of resource discovery systems and networked information retrieval.

General Discussion

The classic vector space model [1] has served very well for the purpose of textual information retrieval. But text is much more than just a set of terms. Firstly, the actual meaning of a word depends on the context in which the word is appearing. Secondly the same words can be combined in different ways to produce texts of different meaning. The classic model takes advantage of redundancy which exists in texts, but how far can we go exploiting this redundancy?

In our view, the main source of improvement of IR efficiency is not in taking similarity measures to perfection, but in using additional information from the text. Good examples of such information are context in which words appear in texts, words order and proximity.

The idea to use this information is not a new one. There is much of research directed at taking advantage of the additional information available in texts. There is also strong evidence that documents can not be treated as homogeneous objects [2]. The ideas of Local Context Analysis (LCA) [3], passage retrieval [2,4] and word sense disambiguation, - all suggest that we are on the right track.

Our own experiments on extended use of context [5] conducted on the Reuters-22173 collection illustrate that use of context can be of a great value, as we achieved very good results using this method. Our first goal in TREC was to test this method on a different collection. To our disappointment, it did not work at all, though, after some consideration we found a possible reason. The main difference between the two collections is that in the Reuters collection all articles are relatively small, and in the TREC collection many articles are large. This means that the assumption that the documents are monocontextual holds more or less for the Reuters collection, but is not true for the TREC collection. Unfortunately, our method depended on this assumption. It was clear that in TREC we had to use proximity information to define context, and we had to devise a method to do that.

The method we finally developed is entirely different from the one described in [5]. It is similar to LCA and passage retrieval, but takes these ideas further. The main idea is that if we have a query and are looking for documents which are the best answers to the query, then 1) the context in which the query words occur in the documents under consideration does matter, and 2) we are only interested in parts of the document which are relevant to this query, the total length of the document is irrelevant (unless we are concerned with the cost of retrieval of very long documents).

In our method, we do not compare a query to a document. We compare context of the query to the context in which query words appear in the document. Comparison of two similar objects (namely, two contexts) seems to be a more reasonable thing to do than to compare two different objects - like a query and a document. Besides, this method seems to model exactly what human beings do when asked a question. If context of the question is unclear, we ask for elaboration; when we believe we have got the context right, we are trying to identify information which has something to do with the context, not necessarily with the exact words of the question. Even if we are asked for very particular information (e.g. a description of event happened on a particular date), we first try to find contextually close information and then filter it to identify items dealing with the particular details we have been asked about.

To perform the task of comparison in our method, we first expand the query to its context in the top relevant documents, exactly as in LCA. Then we reduce every document to be compared to the context of the query (words surrounding the original query words in the document). Then we compare the two contexts.

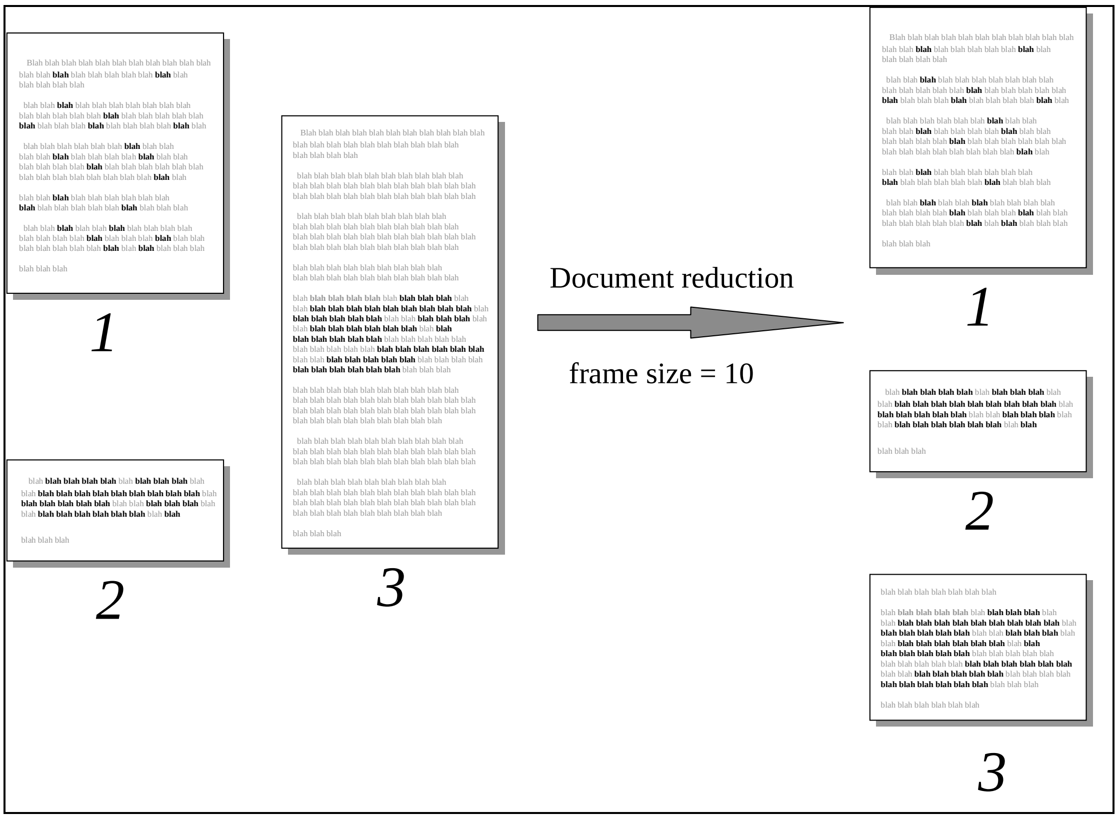

Figure 1. Example of document reduction

The process of reduction is illustrated in Figure 1. Bold parts of text are query terms. Document 1 is irrelevant to the query, but contains accidental query terms. Document 2 is relevant and short. Document 3 contains a relevant part, but is considerably longer than the other two and therefore would be assigned the lowest score of these 3 documents. However, the 3rd document will not be penalized for its length after document reduction. It may well be the case that the documents would be ranked in order 2,3,1. Document reduction facilitates better scoring and provides refined data for training.

We learned that behavior of “document context vectors” (as we call documents reduced to context of a particular query) seems to be quite different from the reported behavior of document vectors. For example, there are only very few of the top positive and negative examples needed to train a classifier. Training on anything but the top scoring documents (both relevant and irrelevant) does not produce any improvement. Negative weights do play their role well. We attribute these differences to the fact that we are dealing with refined material. (At this stage, however, we can not be completely sure of this because the results on the test set were unexpectedly low. A further investigation is required.) By reducing documents to context vectors, we are removing a lot of irrelevant information, and hope not to be removing too much of relevant information. Thus, influence of accidental dependencies is significantly reduced. On the other hand, the core dependencies are enforced.

There is one more advantage of this method which should be mentioned. As it can be seen below in the description of context vectors construction, this method favors documents where the query terms appear close together. For example, it will give higher scores to long documents with a couple of paragraphs containing concentrated query words, than to comparatively short documents where query terms are widely spread - something that the classic vector space model is unable to do.

Document Context Vectors

To construct context vector of a <document, query> pair, we take words of the document appearing within a given range (we call this range "frame size" - FS) of the query terms in the document. The query terms in the document are also included. Note that if occurrence of the query terms in the document is accidental, they will tend to be spread and surrounded by random terms. This will result in a relatively long vector consisting of random terms with low weights. If the document is relevant, the query terms will tend to appear together and surrounded by similar terms (context terms). Frames of the query terms will overlap. It will result in a shorter vector consisting of (context) terms with higher weights.

As it can be seen, we are getting an advantage from two sources. The first source is the consideration of context terms. The second source is penalizing documents where query terms are spread. As we have learnt, these sources have different dependence on the FS parameter. With higher values of FS, more accidental terms are included in the context vectors. It makes context "polluted" and reduces value of the context component. On the other hand, greater value is received from overlapping frames if query terms occur in a close group. This increases value of the proximity component.

Construction of document context vectors is very cheap, especially for systems that support indexing of proximity information.

Query Context Vectors

To construct query context vectors, we use the Rocchio [6] algorithm, but we run it on document context vectors, instead of source documents. We order the resulting vectors by the weights of the words and truncate them to a given number of words (words in vector - WV).

Routing Experiments

Training

To train a classifier for a query, we first ordered all documents according to their relevance scores (top scores first) obtained using the MG query engine [7]. We then used top Npos positive and top Nneg negative examples to train the classifier. We optimized parameter settings using data sets from the previous TREC conferences.

Routing

In the routing stage, all the query vectors were placed in RAM. Every document was read once, a document context vector was created for every query and comparison performed. The best 1000 scores together with document's ids were kept for each query and printed out when all the documents had been processed.

The system was re-trained with new scoring information available and another iteration was performed.

Parameter Settings

Run N1 (automatic):

Iteration 0: MG used to obtain initial scores.
Iteration 1, training: FS = 3, WV = 400, Npos = 10, Nneg = 20, $\alpha = 0$, $\beta = 10$, $\gamma = 4$;
Routing: FS = 3.
Iteration 2, training: FS = 3, WV = 400, Npos = 10, Nneg = 20, $\alpha = 0$, $\beta = 10$, $\gamma = 4$;
Routing: FS = 3.

Run N2 (automatic):
Iteration 0: MG used to obtain initial scores.
Iteration 1, training: FS = 3, WV = 400, Npos = 10, Nneg = 25, $\alpha = 0$, $\beta = 16$, $\gamma = 4$;
Routing: FS = 3.
Iteration 2, training: FS = 3, WV = 400, Npos = 10, Nneg = 25, $\alpha = 0$, $\beta = 16$, $\gamma = 4$;
Routing: FS = 3.

(α, β, γ are the parameters of the Rocchio algorithm.)

Cost

We used a PC with 96Mb of RAM and Pentium II 266MHz processor for these experiments.

Dictionary creation took 20 minutes, training took 1 second per query (on 35 examples), routing run took 0.5 hour per iteration. These times can be significantly reduced in a system that retains proximity information in indexing stage. As our software looses this information, we had to re-scan every document to create document context vector for a query.

Results

Run	at 5 docs	at 10 docs	at 20 docs	at 200 docs	R-precision	Average
N1	0.4894	0.4468	0.3957	0.2080	0.2552	0.2068
N2	0.4936	0.4468	0.3872	0.2078	0.2603	0.2053

Ad-Hoc Experiments

The major difference between routing and ad-hoc query processing is that there is no training data for ad-hoc queries. We have to rely on assumption that the first few top scoring documents are mostly relevant to the query. Given this assumption, we can use these documents as positive examples for building query context vectors, with no negative examples. The rest of the process is identical to routing.

Query Context Vectors Construction

To obtain initial scoring (iteration 0), we assume that the query context vector is the original query. We build a document context vector for every document and obtain a score by comparing it to the query context vector. This process gives slightly better results than the classic vector space model (MG) because it penalizes documents where query terms are widely spread.

We use the top scoring documents and the Rocchio algorithm to construct query context vectors.

Query Processing

The query processing stage is identical to the routing stage in the description of routing experiments above.

Parameter Settings

Run N1 (automatic, full queries):
Iteration 0: query context vector = query, FS = 3;
Iteration 1, training: FS = 10, WV = 400, Npos = 5, Nneg = 0, $\alpha = 0$, $\beta = 1$, $\gamma = 0$;
Processing: FS = 10;
Iteration 2, training: FS = 10, WV = 400, Npos = 6, Nneg = 0, $\alpha = 0$, $\beta = 1$, $\gamma = 0$;
Processing: FS = 10;

Run N2 (automatic, description-only queries):
Iteration 0: query context vector = query, FS = 3.
Iteration 1, training: FS = 17, WV = 400, Npos = 4, Nneg = 0, $\alpha = 0$, $\beta = 1$, $\gamma = 0$;
Processing: FS = 17.

Run N3 (automatic, title-only queries):
Iteration 0: query context vector = query, FS = 3.
Iteration 1, training: FS = 17, WV = 400, Npos = 4, Nneg = 0, $\alpha = 0$, $\beta = 1$, $\gamma = 0$;
Processing: FS = 17.

Cost

We used a PC with 96Mb of RAM and Pentium II 266MHz processor for these experiments.

Dictionary creation took 1.5 hour, training took less than 1 second per query, processing run took 2.75 hour per iteration. These times can be significantly reduced in a system that retains proximity information in indexes.

Results

Run	at 5 docs	at 10 docs	at 20 docs	at 200 docs	R-precision	Average	MG Aver.
N1	0.3360	0.2820	0.2180	0.0756	0.1455	0.1265	0.1208
N2	0.2720	0.2280	0.1840	0.0584	0.1295	0.1171	0.0904
N3	0.3440	0.2860	0.2290	0.0760	0.1481	0.1259	0.1207

Fault Analysis and Conclusion

This method is simple and practical. It demonstrated good performance on the data set which we used for preliminary experiments. However, the routing results on the test data set are disappointing (50% lower in precision). We initially blamed overfitting for this, but later experiments have shown that overfitting is not the cause. Even if we train the system on the TREC-6 routing test set and then run it on the same set, we are still getting significantly lower results than on our training set.

Our analysis has shown that we could improve performance if we had trained the system on all available data, not just on FBIS documents, - something that was hard to do because of our hardware limitations at that time.

We have conducted more ad-hoc experiments after the conference and obtained much better results. We are investigating whether this improvement was a result of different parameters or removing some minor bugs from the software. We intend to publish this work and results later.

We could improve performance if we applied proven techniques commonly used by other groups, but it was beyond the scope of the first year of participation. We feel that we have accomplished our goals for the first year and there is a large amount of work to do ahead.

Acknowledgments

The work reported in this paper has been funded in part by the Co-operative Research Centres Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

References

[1] G. Salton. Automatic text processing: the transformation, analysis and retrieval of information by computer. Addison-Wesley, 1989, 387 p., ISBN 0-201-12227-8.

[2] R. Wilkinson. Effective retrieval of structured documents. In Proceedings, SIGIR 94, pp. 311-317, Dublin, Ireland, 1994. Association for Computing Machinery.

[3] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In Proceedings, SIGIR 96, pp. 4-11, Zurich, 1996. Association for Computing Machinery.

[4] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in information systems. In Proceedings, SIGIR 93, pp. 49-58, New York, 1993. Association for Computing Machinery.

[5] A. Kosmynin and I. Davidson. Using background contextual knowledge for documents representation. In Principles of Document Processing ’96 Workshop, California, 1996.

[6] Jr. J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, The SMART Retrieval System: Experiments in Automatic Document Processing, pp. 313-323, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[7] I. H. Witten, A. Moffat, and T. C. Bell. Managing Gigabytes, Van Nostrand Reinhold, New York, 1994.