Natural Language Information Retrieval: TREC-8 Report

Tomek Strzalkowski
GE Research & Development
Jose Perez-Carballo
Rutgers University
Jussi Karlgren
Swedish Institute of Computer Science
Anette Hulth
Stockholm university
Pasi Tapanainen & Timo Lahtinen
Conexor OY, Helsinki

1. Summary

This report describes the adhoc experiments performed by the GE/Rutgers/SICS/SU/Conexor team in the context of TREC-8. The research efforts went in four directions:

1. As in previous years, we performed a full linguistic analysis of the entire corpus, and used the results of the analysis to provide index terms on a higher level of abstraction than can be provided by stems alone.
2. We made use of two different query expansion techniques, one automatic and one manual, both developed for TREC-8.
3. The various analysis models were combined using a stream model architecture, where each stream represents an alternative text indexing method, and the stream's various overlapping knowledge was merged using a new merging algorithm derived from first principles.
4. The entire text was analyzed for various stylistic items.

Due to the distributed approach, this years' research efforts partly canceled out each other. New experiments in every step of the process did not result in an overwhelming overall result. We are able to determine that the manual query expansion technique developed at General Electric performed very well.

2. Background

The work reported here was part of the Natural Language Information Retrieval project (NLIR) (Perez-Carballo, Strzalkowski, 1999; Strzalkowski et al., 1998 and 1997; Strzalkowski, 1995). One of the thrusts of this project has been to demonstrate that robust NLP techniques can help to derive better representation of text documents for indexing and search purposes than any simple word and string-based methods commonly used in statistical full-text retrieval. This was based on the premise that linguistic processing can uncover certain critical semantic aspects of document content, something that simple word counting cannot do, thus leading to a more accurate representation. In earlier experiments we demonstrated that NLP could be done efficiently on a very large scale, and that it could have a significant impact on the performance of the IR systems we were using then. At the same time, it became clear that exploiting the full potential of linguistic processing is harder than originally anticipated. In particular, simple linguistically motivated indexing (LMI) techniques turned out to be no more effective than well-executed statistical approaches , at least for English , while more advanced NLP techniques, such as concept extraction, remained too expensive for large-scale applications (Sparck-Jones, 1999).

Given this state of affairs, we went on to investigate specific conditions under which LMI could be more beneficial. For example, we have noticed that the amount of improvement in recall and precision which we could attribute to NLP, appeared to be related to the type and length of the initial search request. Longer, more detailed topic statements responded well to LMI, while terse one-sentence search directives showed little improvement. This is not particularly surprising considering that the shorter queries either contain a handful of highly discriminating terms or are deliberately vague. We adopted the topic expansion approach in which the original topic is expanded using passages selected from sample retrieved documents. The intent was to expand the initial search specifications in order to cover their various angles, aspects and contexts. Based on the observations that NLP is more effective with highly descriptive queries, we designed an expansion method in which passages from related, though not necessarily relevant documents were imported into the user queries. This method produced a fairly dramatic improvement in the performance of several different statistical search engines that we tested boosting the average precision by anywhere from 40% to as much as 130%. Therefore, we concluded that topic expansion appears to lead to a genuine, sustainable advance in IR effectiveness. Moreover, we showed in TREC-7 that this process can be automated while maintaining at least some of performance gains. Thus far we have used only very simple linguistic tools (i.e., those suitable for high-volume IR applications) to assist automatic expansion, but we see this area as ripe for more advanced processing techniques, including entity and event extraction, co-reference and cross-reference techniques, etc.

3. Processing scheme

InQuery was used as the indexing and retrieval engine. This year, the linguistic processing of TREC data, both text and queries, was performed in Helsinki using the newly developed Functional Dependency Grammar (FDG) text processing toolkit. The processed text was sent via ftp to Rutgers and SICS for indexing.

For some of the manual submissions, the topics were processed at General Electric using the interactive Query Expansion Tool for manual query expansion; for the automatic submissions, queries were expanded at Rutgers using a passage retrieval algorithm. The expanded topics were processed in Helsinki to obtain matching search terms for the linguistic indices, and sent back to Rutgers for retrieval.

The results from the various processing and retrieval streams were merged to obtain a final rank order using a merging algorithm developed for this years' TREC at Stockholm university and SICS.

4. Streams

The stream model (see Fig. 1) has been described in previous TREC papers and in (Perez-Carballo, Strzalkowski, 1999). Each "stream" uses its own index which is created using a different indexing technique (some of them involving linguistic processing). The results obtained from all the streams are combined using merging algorithms. In past TRECs we were able to obtain significant improvements in performance over the baseline single word indexing stream. The experiments for TREC-8 did not yield similar improvements. This year, in spite of improved analysis machinery, the linguistic streams performed far less well than earlier years and we were not able to combine them usefully with the standard retrieval streams. There are several possible explanations, but the main reason appears to be the several simultaneous changes in our approach. We changed the character of the streams, reworked the merging algorithm, and as a result we have not been able to make use of previous years' experience in matching query processing with text processing and combining results appropriately in time for this report. We are continuing the experiments that we expect to publish elsewhere.

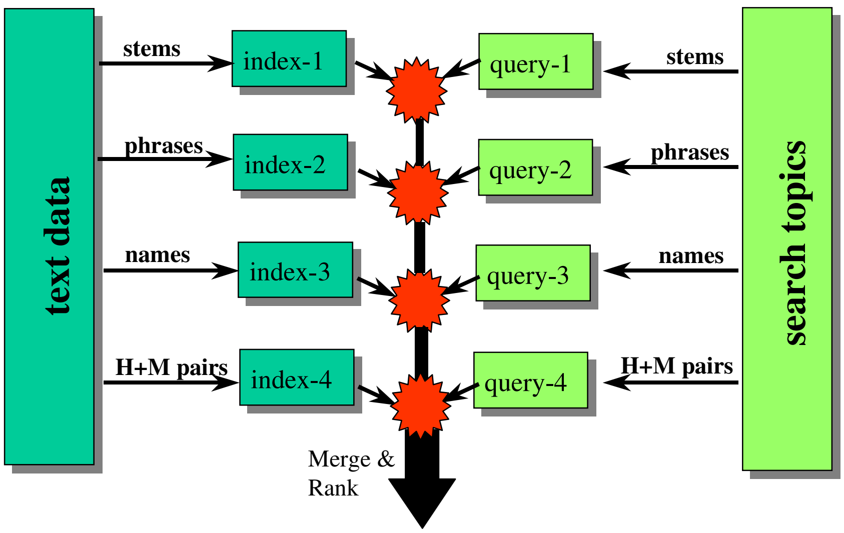

Figure 1. Stream Model organization

4.1 Linguistic Streams

Some of the linguistic streams we used were created using Helsinki's Functional Dependency Grammar (FDG) which includes the EngCG-2 tagger and dependency syntax which links phrase heads to their modifiers and verbs to their complements and adjuncts. FDG was applied to the whole corpus, with the output passed to the stream extractor.

We tried to merge the results obtained from linguistic streams with the stems stream, as we have done other years, but were unable to obtain good results (i.e. improve the performance of the stem stream). Because of lack of disk space we could not use the same automatic expansion algorithm on the linguistic streams so we cannot draw any conclusions yet about linguistic streams or the merging algorithms.

In one experiment we used the InQuery #phrase (see below) operator in order to add phrases from one of the linguistic streams to the query generated using manually chosen summaries. This seemed to actually decrease the performance of that run.

In order to have a baseline to be used with the linguistic streams we added InQuery's #phrase operator to words that appeared close to each other in the topics. No linguistic processing was used at all. This was done automatically. Some of the “phrases” obtained did not seem to make any sense and no human would have added them. Surprisingly, the queries that used that device performed better than the linguistic streams and in some cases better than the pure stems (some of our “official” runs, reported below, use this technique). We are now performing further experiments and tests. More definitive results and analysis will be presented in the final form of this paper.

InQuery's #phrase operator (quoting from the InQuery manual):

```
 Phrase Operator: #phrase(T1 ... Tn)

 Terms within this operator are evaluated to determine if they occur
together frequently in the collection. If they do, the operator is
treated as an ordered distance operator of 3 (#od3). If the arguments
are not found to co-occur in the database, the phrase operator is
turned into a SUM operator. In ambiguous cases the phrase becomes the
MAX of the SUM and the OD3 operators.
```

5. Merging the streams

In previous years, the merging algorithm for TREC was tuned through trial and error, and we always managed to find relative weighting that improved the score. This process is of course unsatisfactory and we tried to capture some dependencies that would help to automatically estimate the merging function. For the past two TRECs we added a rank dependent non-linear weighting scheme in which a document’s final rank in the merged ranked list is a function of average precision of component streams, as well as of the rank of the document rank in the various streams. This change had a positive effect on merging precision, but it still required supervised training in order to optimize the parameters.

This year, we decided to use a more principled approach. We performed a set of merging experiments using some streams that we judged the most promising based on early experiments. For lack of time and processing space, and trivial file transfer problems, we restricted the experiments to the following four streams, postponing the inclusion of much of the linguistic and stylistic experiments made during the course of the project:

1. run.7.proc.PH.t3d1n1.35: automatic expansion, proximity phrases, words from title are repeated 3 times, runs on *stem* stream.
2. run.7.ph.t3d1n1.35: before expansion the terms from the *ph* stream are added to the topics using the #phrase operator in case it is a phrase, automatic expansion, words from title are repeated 3 times, runs on *stem* stream.
3. run.7.proc.P.t3d1n1.35: automatic expansion, words from title are repeated 3 times, runs on *stem* stream.
4. run.7.thr.t3d1n1: words from title are repeated 3 times, runs on the *thr* stream.

Training several different classifiers and combining the predictions of these into a single prediction is a common method for creating an accurate classifier from a set of training data (Breiman, 1996; Drucker, et al, 1994; Wolpert, 1992). A number of researchers have demonstrated that such combined classifiers in general are more accurate than any of the constituent classifiers (Dietterich, 1997; Breiman, 1996; Merz, 1999; Quinlan, 1996; Wolpert, 1992; Zhang, Mesirov & Waltz, 1992). Bartell, Cottrell and Belew (1994) have shown similar results in the document retrieval domain: using different retrieval algorithms and then combining them may significantly improve retrieval performance.

The streams in our model capture different aspects of the documents' content. When merging the streams, the aim is to produce a final result that is more accurate, i.e., has a higher average precision, than the output of any of the individual streams. The final result should therefore be a richer set of documents. Obtaining this result is, however, not trivial.

As detailed above we compose a mixture of different indexing approaches, term extracting, weighting strategies, and different search engines we use into indexing streams. Each stream represents an alternative text indexing method; some require complex linguistic processing, while others are based on simple quantitative techniques. The results obtained from the different streams are lists of documents ranked in order of relevance: the higher the rank of a retrieved document, the more relevant it is presumed to be (in comparison to the other retrieved documents). The ordering is based on the relevance score - a figure produced by the stream, reflecting the document's accuracy as judged by the system. The streams perform in parallel and the results from the different streams should be merged to produce one final result. As the streams capture different aspects of the documents' content, the final result should be a richer set of documents. The aim of the merging is to produce a final result that is more accurate than the output of any of the individual classifiers. Obtaining such a result is, however, not trivial.

A merging algorithm called SEQUEL (Asker & Maclin, 1997) was implemented for the task. The rationale behind SEQUEL is to find the most confident classifier down to a certain threshold. It requires that the lists are sorted by - in this case - relevance score. The confidence is calculated by finding the classifier with the highest proportion correct classifications (i.e., relevant documents) at the top, down to the first non-relevant one. The threshold will be the lowest relevance score within this interval. The items covered by the span are removed from all classifiers. At a certain value the best performing classifier is considered the default classifier, i.e., it is used as the single classifier.

The algorithm was trained on 40 out of 50 queries – setting aside 10 queries for testing – from the TREC-7 materials, using the TREC-7 data and relevance judgments. All non-judged documents were removed, leaving only the judged documents for consideration. Two different implementations of the algorithm were made: one where all queries were sorted by the judgment of the system; and one where the program examined the confidence for each query at the time, taking the average as the result.

Although the algorithm performed well, the combined classifier did not beat the best individual classifier. This could possibly be because of the algorithm not being very “forgiving”: immediately upon finding an irrelevant document the stream is discarded. SEQUEL also tended to work with “chunks” of documents, covering too many at the time. This could be due to the fact that the relevance scores given by the retrieval systems range over a quite limited span. (An implementation that normalized the ranking scores to fall between 1 and 0 was also made, but the improvement was not significant.)
The algorithm tended to favor the best performing classifier (it being the most confident stream) and discard the additional information that the weaker streams may contribute with. Neither does the method take a possible overlap of retrieved documents in different streams into account. The algorithm was implemented to consider the top 1000 for each question and classifier. This means that for the majority of the documents, we only had judgments from one or two streams. It would be more appealing to apply a method where every document could take advantage of the fact that we use several different retrieval methods.

6. Automatic expansion algorithm description.

Using the automatic expansion algorithm described in this section we obtained a 37% improvement of average precision over a baseline where no expansion was used.

6.1 Algorithm:

1. The topics sent by NIST were processed to eliminate some words and phrases such as: “a relevant document”.
2. The title text was repeated 3 times, the description 2 times. Our intent was to give different weights to the different fields.
3. Processed topics were submitted to InQuery in order to retrieve the top 20 documents.
4. For each one of the D documents with highest document score larger than threshold T, extract all passages of size larger than S. Let the passage score be the sum of all unique occurrences of a query term (either word of phrase) in that passage.
5. Choose the P passages with highest passage score and add them to the original query. Notice that given two passages A and B, the score of B may be higher than A (and thus B may be chosen over A) even though A may belong to a document that has a higher score than B's document.

The values used for the parameters described above were: D = 5; T = .432; S = 50; P = 12

7. Ad-Hoc submissions

We submitted 4 runs for the ad-hoc track.

query id	Relevant documents retrieved out of 4728	Average precision	Precision at 10 documents	R-Precision	query description
1. 8manexT3D1N0 (judged)	0.3325	0.3346	0.6520	0.3671	manually-assisted topic expansion using only title and description fields.
2. GE8ATDN1 (judged)	0.3138	0.2623	0.5020	0.2984	automatic topic expansion using title description and narrative fields. Proximity phrases only.
3. GE8ATDN2 (not judged)	0.3068	0.2580	0.5498	0.2993	automatic topic expansion using title description and narrative fields. No phrases used.
4. GE8ATD3 (not judged)	0.3022	0.2618	0.5658	0.2959	automatic topic expansion using only title and description fields. Proximity phrases plus original text.

Below are short descriptions of our official ad-hoc runs (shown in the table above). All official runs submitted were produced using only the stem stream as opposed to being the result of merging evidence from different streams.

7.1 Summarization-based manually-assisted topic expansion run (8manexT3D1N0)

Manually-assisted topic expansion using only title and description fields. The methods used to obtain this run where the same as the ones we used in TREC-7. Summaries used in expansion were derived from top-ranked documents retrieved by SMART using the initial topics (title+description only). The key characteristics of this run is the 10 minute time limit imposed on topic expansion. All expansion has been performed via the Query Expansion Tool interface (QET) which allows the user to view only the summaries of top retrieved documents, and select or deselect them for topic expansion. By default, summaries of all top 30 documents were used for expansion unless the user manually deselected some (this was precisely the only form of manual intervention allowed.) We observed that for many queries 2 interactions were possible within the 10 minute interval. The first interaction (submit original query, wait for result, get 30 summaries, review & deselect summaries, and commit the selections) would take typically 4-6 minutes. In the second interactions, only the new documents retrieved in top 30 ranks (if any) were considered, therefore usually 3-4 minutes were sufficient. The target of expansion was to get between 5 and 10 "relevant" summaries within the allotted time. If this was achieved within the first interaction, no further search was performed. Otherwise, the second interaction was attempted if at least 3 minutes remained. This 6-4 split was determined in dry-run trials with TREC-6 queries. The topic expansion interaction proceeds as follows:

1. The initial natural language topic statement is submitted to a standard retrieval engine via a Query Expansion Tool (QET) interface. The statement is converted into an internal search query and run against the database.
2. The system returns topic-related summaries of top N (=30) documents that match the search query.
3. The user reviews the summaries (approx. 5-15 seconds per summary) and de-selects these that are not relevant. For TREC-8 evaluations (like for TREC-7), we set a time limit of 10 minutes per query (clock time).
4. All remaining summaries are automatically attached to the original search topic.
5. bThe expanded topic is passed through a series of natural language indexing steps and then submitted for the final retrieval.

7.2 Automatic topic expansion run (GE8ATDN1)

This run uses title description and narrative fields. Proximity phrases only: the query text is replaced by the output of an algorithm that linked words which appeared in the same sentence at less than 3 words of each other using the #phrase operator. This run was intended as a baseline but became an official run when we were unable to beat its performance.

7.3 Automatic topic expansion run (GE8ATDN2)

This run uses title description and narrative fields, like GE8ATDN1, but no phrases are used.

7.4 Automatic topic expansion run (GE8ATD3)

This run, again, uses only title and description fields. It uses "proximity phrases" formed out of words occurring together.

8. Continuing work

We are currently performing a set of new experiments that takes into account the rankings obtained from every stream. All documents that have been ranked among the first 1000 documents by at least one stream out of N streams constitute a "document pool" of at least 1000 documents and not more than N*1000 documents. We let every stream score all documents in this pool. As a first experiment, we will implement a simple linear combination of the judgments from the four streams. A weighted sum of all the scores from each of the streams will get us the total score that includes the knowledge of all streams in question. As mentioned before, the span for the ranking scores is not that large, and therefore even a very small score can alter the ordering of the documents. For these tests we will retain the non-judged documents.

There is a possibility that some documents could get a better total score by having been given four low scores (too low to be among the best 1000 documents for any stream) than one with a high score on one and no rating on the other streams. However, if none of our streams ranks a document among the top 1000 we will discard it. If the experiments with simple linear combinations turn out satisfactory – i.e. better than the non-adapted learning algorithms – we will continue with more sophisticated methods, by for example weighting the different streams.

The main point with more "forgiving" classifiers, is to not discard a stream immediately upon finding an irrelevant document: document relevance is a debatable issue in itself, and cannot easily be compared to other classification tasks where the errors are of a more clear-cut nature.

8.1 Further experiments

This paper is just a preliminary report. Before we present a final version for the TREC-8 proceedings we must complete at least the following experiments:

- Create indexes for all linguistic streams that allow for query expansion using the same algorithm used for the *stem* stream. Try merging algorithms again but with runs obtained using query expansion.
- Merge: *stem* run (without expansion) with some (or each of the) linguistic stream(s) (without expansion). Can we get any improvement from merging? If yes, then expansion is drowning the improvement obtained by linguistic streams.
- Compare performance of automatic expansion using passages and automatic expansion using summaries.
- Compare the performance of runs obtained using LMI vs. runs obtained linguistic processing on the queries only.

9. Conclusions

Preliminary results seem to suggest it would be possible to get as good a performance by processing the query (including expansion) and using an IR system with an expressive query language such as InQuery as what we get creating indexes using sophisticated and very expensive linguistic techniques. We should explore the possibilities of using a much more sophisticated linguistic analysis of the queries but less on the index.

10. References

Asker, L. & Maclin, R. (1997). Ensembles as a Sequence of Classifiers. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97.

Bartell, B.T., Cottrell, G.W. & Belew, R.K. (1994). Automatic Combination of Multiple Ranked Retrieval Systems. In: Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval.

Breiman, L. (1996). Bagging predictors. Machine Learning, vol. 24(2) pp. 123-140.

Callan, Jamie. 1994. “Passage-Level Evidence in Document Retrieval.” Proceedings of ACM SIGIR'94. pp. 302-310.

Clemen, R. (1989). Combining forecasts: A review and annotated bibliography. International Journal of Forecasting, vol. 5 pp. 559-583.

Drucker, H., Cortes, C., Jackel, L., LeCun, Y., and Vapnik, V. (1994). Boosting and other machine learning algorithms. In: Proceedings of the Eleventh International Conference on Machine Learning. Morgan Kaufmann.

Jarvinen, T., and Tapanainen, P. 1997. “A dependency parser for English.” Tech. Rep. TR-1, Department of General Linguistics, University of Helsinki, Finland.

Jarvinen, T., and Tapanainen, P. 1998. “Towards an implementable dependency grammar.” In Processing of Dependency-Based Grammars. Montreal, Canada. S. Kahane and A. Polguere, Eds., COLING-ACL'98, Association for Computational Linguistics, Universite de Montreal, pp. 1-10.

Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan. 1993. “Retrieval Experiments with a Large Collec-tion using PIRCS.” Proceedings of TREC-1 conference, NIST special publication 500-207, pp. 153-172.

Merz, C. J. (1999). Using Correspondence Analysis to Combine Classifiers. Machine Learning, vol. 36(1/2) pp. 33-58.

Perez-Carballo, Jose, Strzalkowski, Tomek (1999) Natural language information retrieval: progress report. Information Processing and Management, Vol. 36, No. 1, pp. 155-178. Pergamon/Elsevier.

Quinlan, J. R. (1996). Bagging, boosting, and c4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence.

Sparck-Jones, Karen. 1999. “What Is The Role for NLP in Text Retrieval”. In T. Strzalkowski (ed.) Natu-ral Language Information Retrieval. Kluwer. pp. 1-25.

Strzalkowski, T., Stein, G., Wise, G.B., Perez-Carballo, J., Tapanainen, P., Jarvinen, T., Voutilainen, A. & Karlgren, J. (1998). Natural Language Information Retrieval: TREC-7 Report. In: Proceedings of the Seventh Text REtrieval Conference (TREC 7). NIST Special Publication, Gaithersburg: NIST.

Strzalkowski, Tomek, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. “Natural Language Information Retrieval: TREC-6 Report. ” Proceedings of TREC-6 conference.

Strzalkowski, Tomek, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, Jose Perez-Carballo, Troy Straszheim, Jin Wang, and Jon Wilding. 1997. "Natural Language Information Retrieval: TREC-5 Report." Proceedings of TREC-5 conference.

Tapanainen P & Järvinen T, 1997, A non-projective dependency parser. ANLP'97, p. 64-71, Washington DC.

Tapanainen P., 1999, Parsing in two frameworks: finite-state and functional dependency grammar. Ph.D. thesis, Language technology, University of Helsinki.