JHU/APL at TREC 2004:
Robust and Terabyte Tracks

Christine Piatko, James Mayfield, Paul McNamee, and Scott Cost
Research and Technology Development Center
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road, Laurel, Maryland 20723-6099 USA
{Christine.Piatko, James.Mayfield, Paul.McNamee, Scott.Cost}@jhuapl.edu

Overview

The Johns Hopkins University Applied Physics Laboratory (JHU/APL) focused on the Robust and Terabyte Tracks at the 2004 TREC conference.

For initial ranked retrieval, we continue to use a statistical language model to compute query/document similarity values. Hiemstra and de Vries [3] describe such a linguistically motivated probabilistic model and explain how it relates to both the Boolean and vector space models. The model has also been cast as a rudimentary Hidden Markov Model [4]. Although the model does not explicitly incorporate inverse document frequency, it does favor documents that contain more of the rare query terms. The similarity measure can be computed as

$$Sim(q,d) = \prod_{t \in q} \left(\alpha \cdot f(t,d) + (1-\alpha) \cdot f(t,C) \right)^{f(t,q)}$$

Equation 1. Similarity calculation.

where α is the probability that a query word is generated by a document-specific model, and *(1- α)* is the probability that it is generated by a generic language model. *f(t,C)* denotes the mean relative document frequency of term *t*. We have observed that aggregate performance using this model is fairly insensitive to the precise value of α that is used; however, higher values of alpha tend to result in selecting documents that contain a greater number of the query terms.

Robust Track

In last year's TREC Robust Track, we investigated merging (many) disparate run files, using automated techniques such as SVM classification to create a single, robust run. While we had modest success improving our runs over our baseline, we did not approach the theoretical maximum of an oracle choosing the best run per query.

We examined about half of the 50 difficult queries from TREC 2003 by hand (similar to an effort done by a larger team [1, 2]). Our findings were much the same.

We observed that relevant documents for these difficult queries did contain concepts related to each title word. In addition, these title concepts could be found "fairly close together" in the relevant document. For difficult queries, our system sometimes discounted one of the title concepts (it was not found at all in a highly ranked document). Other times it found documents with most of the terms, but widely scattered in the returned documents.

For example, for a query such as "Hubble Telescope Achievements," both query expansion and term weighting seemed to reduce the importance of the concept of "Achievements." In fact, one of the relevant documents has the phrase "achievements of the US Hubble Space Telescope," but for our system was not always a top scoring document.

We also noted, even for difficult queries, it seemed that a good fraction of relevant documents did appear top 1000 documents, they were just not ranked highly enough. We thus chose to focus this year on ways to rerank an existing run to try to improve performance. We focused on boosting documents with more title concepts appearing closer together.

We did not make use of *any* external resources, such as the Web, which were shown to be quite beneficial in the TREC-2003 Robust Track.

We reused indices from last year that used various tokenization methods. Summary information for the indices that we used is shown Table 1.

Table 1. Index Statistics for the Robust Track Collection

		# Terms	Index Size
words	w	554751	373 MB
stems (Snowball)	s	455803	320 MB

Minimal Matching Span

We opted to try applying the Minimal Matching Weighting of Monz to re-rank, hoping to improve our typical good runs by favoring documents with more query terms appearing closer together.

Monz applied this scoring method to improve QA performance in his thesis work [5], since he had not seen much benefit using fixed length overlapping passages for retrieval to improve QA performance. The Minimal Matching Weighting score is a linear combination of the retrieval system score (in our case, a scaled language model score) with a Minimal Matching Span Score, related to the number of matching terms in the document and the length of the closest span in which they appear together.

Roughly speaking, the minimal matching span (MMS) of a set of terms is the minimal length of a consecutive set of document terms containing at least one occurrence of each term in the set.

If there is more than one matching query term in the document, the new minimal span weighting score (MSW) is computed by interpolating between a weighted version of minimal matching span of matching query terms and the normalized language model score (S). If there are q terms in the query and $q_{matching}$ is the set of query terms that appear in the document, MSW is as follows:

$$MSW = \lambda S + (1 - \lambda) (|q_{matching}|/MMS)^{\alpha} (|q_{matching}| / |q|)^{\beta}$$

Equation 2. Monz Minimal Span Weighting Score

Monz empirically determined parameters $\lambda = 0.4$ $\alpha = 1{,}8$; $\beta = 1$ based on TREC-9 data, and we reused these values for most runs (we used $\lambda = 0.5$ for a TDN run based on performance on TREC2003 data).

Prior to submission we estimated our performance examining performance on the previous queries used in the TREC 2003 evaluations. We observed some benefits to reranking using this rescoring Minimal Matching Weighting, using matching title words (or stems) for both title-only and TDN runs for various robust measures, and this was confirmed in our officially submitted runs (see the Robust Runs Performance section below).

We focused on improving the hardest topics, as suggested in the Robust Track guidelines, since MAP-Hardest is most affected by the most difficult topics. We did not focus on high mean average precision (averaged over all topics) in our base runs and primarily concerned ourselves with improvements of robust measures of performance.

Robust Runs

We used five baseline runs (not submitted): **Ts** (title-only, stem index), **Tw** (title-only, word index), **D** (desc-only, word index), **TDN** (title-desc-narr, word index), and **TDNf** (title-desc-narr using relevance feedback, word index). For each baseline run, we normalized the language model scores of the top 1000 documents retrieved to between 0 and 0.9. We submitted five runs, reranking each of the baseline runs.

apl04rsTs (***Tsr***) is the combination of two runs, using the stem index, the (normalized as described above) language model scores, the Title topic field only, and this run reranked with minimal matching span using the Title only. No relevance feedback was applied.

apl04rsTw (***Twr***) is the same as above, using the word index.

apl04rsDw (***Dr***) is the combination of two runs, using the stem index, the (normalized as described above) language model scores, the Title topic field only, and this run reranked with minimal matching span using the Description only. This was our mandatory description-only run. To perform better we should have chosen a subset of "important" words from the description. No relevance feedback was applied.

apl04rsTDNw5 (***TDNr***) is a combination of two runs, one using the words index and the (normalized as described above) language model scores on the Title, Description and Narrative topic fields and this run reranked with minimal matching span using the Title only. No relevance feedback was applied, and for this run only λ was chosen based on the previous TREC data to 0.5.

apl04rsTDNfw (TDNfr) is a combination of two runs, one using the words index and the (normalized as described above) language model scores on the Title, Description and Narrative topic fields. Relevance feedback was applied. This run was reranked with minimal matching span using the Title only.

Robust Runs Performance

Overall, we observed modest improvements in most robust measures using the reranking approach. Below are tables of performance of officially submitted runs for various subsets of topics. In each of the tables, grey boxes indicate where the measure improves after using minimal span reranking.

The tables below generally show improvement for the measures of mean average precision (MAP), precision at 10 (P(10)), and Area (which is described in the Robust Track Overview). The largest and most dramatic increases are in precision at 10.

Table 2. Effect of Span Reranking using T on **Tw** (word) Run

Topic Set	**MAP**		**P (10)**		**# no-rel@10**		**Area**	
	Tw	***Twr***	**Tw**	***Twr***	**Tw**	***Twr***	**Tw**	***Twr***
200 old topics	.1970	*.2078*	.3303	*.3815*	25 (12.5%)	*29 (14.5%)*	.0086	*.0092*
49 new topics	.2366	*.2462*	.3252	*.3571*	6 (12.2%)	*6 (12.2%)*	.0136	*.0166*
50 hard topics	.1031	*.1107*	.2173	*.2660*	9 (18.0%)	*8 (16.0%)*	.0062	*.0063*
249 all topics	.2048	*.2154*	.3293	*.3767*	31 (12.4%)	*35 (14.1%)*	.0090	*.0101*

Table 3. Effect of Span Reranking using T on ***Ts*** (stem) Run

Topic Set	**MAP**		**P (10)**		**# no-rel@10**		**Area**	
	Ts	***Tsr***	**Ts**	***Tsr***	**Ts**	***Tsr***	**Ts**	***Tsr***
200 old topics	.2207	*.2388*	.3480	*.4080*	23 (11.5%)	*27 (13.5%)*	.0105	*.0129*
49 new topics	.2566	*.2701*	.3374	*.3857*	6 (12.2%)	*5 (10.2%)*	.0122	*.0209*
50 hard topics	.0947	*.1125*	.2027	*.2640*	9 (18.0%)	*7 (14.0%)*	.0066	*.0086*
249 all topics	.2278	*.2449*	.3459	*.4036*	29 (11.6%)	*32 (12.9%)*	.0104	*.0137*

Table 4. Effect of Span Reranking using D on **D** (word) Run

Topic Set	**MAP**		**P (10)**		**# no-rel@10**		**Area**	
	D	***Dr***	**D**	***Dr***	**D**	***Dr***	**D**	***Dr***
200 old topics	.1990	*.1915*	.3243	*.3510*	23 (11.5%)	*30 (13.5%)*	.0071	*.0067*
49 new topics	.2500	*.2373*	.3293	*.3633*	3 (6.1%)	*4 (8.2%)*	.0203	*.0221*
50 hard topics	.1031	*.1073*	.2133	*.2640*	9 (18.0%)	*8 (16.0%)*	.0053	*.0054*
249 all topics	.2091	*.2006*	.3253	*.3534*	26 (10.4%)	*34 (13.7%)*	.0078	*.0077*

Table 5. Effect of Span Reranking using T on **TDN** (word) Run

Topic Set	**MAP**		**P (10)**		**# no-rel@10**		**Area**	
	TDN	***TDNr***	**TDN**	***TDNr***	**TDN**	***TDNr***	**TDN**	***TDNr***
200 old topics	.2608	*.2768*	.4210	*.4970*	9 (4.5%)	*12 (6.0%)*	.0200	*.0219*
49 new topics	.3017	*.3075*	.4054	*.4490*	2 (4.1%)	2 (4.1%)	.0503	*.0587*
50 hard topics	.1370	*.1526*	.2947	*.3780*	1 (2.0%)	*4 (8.0%)*	.0127	*.0122*
249 all topics	.2689	*.2828*	.4179	*.4876*	11 (4.4%)	*14 (5.6%)*	.0227	*.0260*

Table 6. Effect of Span (0.5) Reranking using T on **TDN** (word) Relevance Feedback Run

Topic Set	**MAP**		**P (10)**		**# no-rel@10**		**Area**	
	TDNf	***TDNfr***	**TDNf**	***TDNfr***	**TDNf**	***TDNfr***	**TDNf**	***TDNfr***
200 old topics	.2936	*.3078*	.4490	*.5100*	22 (11.0%)	*23 (11.5%)*	.0149	*.0208*
49 new topics	.3720	*.3557*	.4313	*.4837*	2 (4.1%)	*1 (2.0%)*	.0617	*.0697*
50 hard topics	.1461	*.1618*	.3053	*.3620*	8 (16.0%)	*10 (20.0%)*	.0089	*.0105*
249 all topics	.3091	*.3172*	.4455	*.5048*	24 (9.6%)	*24 (9.6%)*	.0187	*.0255*

Given that the increases are mainly in precision at 10, it would be interesting to incorporate this approach with one round of relevance feedback. Selecting key description terms (as opposed to all non stop words) would also improve the performance of description-only reranking.

Table 7. Comparing TDN Results to Median

250 topics	**Median AP**			**MAP**	**#(%)no-rel@10**	**Area**
Runtag	**Best**	**(>/=/<)**	**Worst**			
TDNr	1	146/3/100	0	.2828	14 (5.6%)	0.0260
TDNfr	8	173/0/76	0	.3172	24 (9.6%)	0.0255

We did not make use of our aggressive run combination approach from TREC-2003 to get the best possible baselines for our T and D runs, so those runs remained roughly median. It will be interesting to try our reranking technique on top title-only TREC 2004 submissions to see if the approach still provides any boost. Our TDN runs (see Table 6) did compare reasonably well to all submissions, and reranking still showed measurable improvements, particularly for precision at 10, so we are optimistic the technique will apply even with higher baseline runs.

We feel our preliminary results show the value of reranking favoring query concept terms appearing closer together in top-ranked documents. Further experiments are needed on a wider variety of base runs to confirm the general applicability of this approach.

Robust Topic Difficulty Prediction Results

We used an average span statistic of top documents as an estimate of topic difficulty. We averaged just the minimal span part of the Monz score (without the normalized language model score interpolation) over the top 10 documents. These scores were then sorted to produce the topic ranks required by the Robust Track. We did not use this statistic for prediction in our runs. We observed only a very weak correlation between topic hardness and this average span statistic.

Table 8. Hardness Correlation using Average Top-10 Span Statistic

Runtag	**Kendall correlation**
Ts	0.200
Tw	0.172
D	0.178
TDNr	0.162
TDNfr	0.175

Terabyte Track

Given the difficulties of indexing a collection as large as the TREC terabyte collection, we were interested in performing the terabyte evaluation without indexing the collection. That is, in as few passes over the data as possible we sought to score every document on every query without building any index structures. This effectively treats the task as a routing problem. It makes sense to index a smaller collection if it will be searched more than a few times. However, on a Terabyte scale collection, when evaluating hundreds of queries in parallel, the point at which it makes sense to build an index is not so clear.

Our approach is to reduce the set of queries to a deterministic finite-state automaton (DFA) that processes text one character at a time and identifies each occurrence of each query term in the collection. First, we generate the cross-product of the following sets:

- the fifty topics for the track;
- five query length combinations: topic-only, topic and description, topic, description and narrative, description-only and narrative-only; and
- words, character 4-grams and character 5-grams.

This generates 750 queries from the fifty topics. Next, we build a non-deterministic finite-state automaton (NDFA) that recognizes each term in each query as it is fed characters from the document stream. This NDFA is then converted to a DFA. We now have a deterministic way to identify each occurrence of each query term. The DFA is run over each document in the collection. The document is scored, and the result is placed in the appropriate heap (each query maintains its own heap of its top scoring documents). Term statistics are required to calculate the language model similarity metric; we calculated term statistics over a small portion of the collection for this purpose.

We implemented the system in Perl, using the PerlIO::gzip package to uncompress the data on-the-fly. We checkpointed the results every so often to guard against system crashes. The system ran on two Sun systems, each with 4G of main memory, using three CPUs on each. Unfortunately, serious system problems prevented us from processing the entire collection by the submission date; we processed only 1% of the collection. The system has a larger memory footprint than we had anticipated. While no significant memory allocation is performed in the portion of the code that we wrote, it is possible that the package we are using to uncompress the data uses memory dynamically.

After the submission deadline, we ran the system on the entire collection. Mean average precision for the resulting runs are shown in Table 9. These runs use no blind relevance feedback; we expect that the scores could be significantly improved with such feedback, but doing so would entail a second pass over the data.

Table 9. Mean average precision for Terabyte Track runs

	4-grams	**Words**
T	13.08	16.14
TD	18.72	21.99
TDN	25.69	29.18
Merged TDN	29.89	

Conclusions

Our Robust Track results give some evidence that reranking of our runs, using Monz's Minimal Matching Span scores, improves robustness. This benefit needs to be confirmed on other systems. For example, we would like to try reranking other systems' runs from the TREC 2004 Robust Track, and confirm robustness performance improvements on these runs as well.

We hope to expand our reranking approach to use more general concept matching, instead of exact title words or stems. A similar idea of *conceptual indexing* was explored by Sun researchers [6]. We began some initial experiments with title word expansion using WordNet that did not complete in time for official submissions. However we do believe title (or short query)-to-concept expansion and reranking to favor documents having more concepts appearing closer together could further improve robustness.

The routing approach to retrieval over the Terabyte collection is attractive, in that it scales linearly with the collection size, and consumes no additional disk resources. It can handle many queries in parallel, and in theory can maintain a fixed memory footprint. Unfortunately, Perl seems not to have been kind to this approach. We suspect that recoding in a lower level language might ameliorate some of these difficulties.

References

[1] C. Buckley. 'Why current IR engines fail.' Proceedings of the 27th International Conference on Research and Development in Information Retrieval (SIGIR-04), pp. 584-585, 2004.

[2] C. Buckley and D. Harman. 'Reliable Information Access Final Workshop Report,' http://nrrc.mitre.org/NRRC/Docs_Data/RIA_2003/ria_final.pdf, 2003.

[3] D. Hiemstra and A. de Vries. 'Relating the new language models of information retrieval to the traditional retrieval models.' CTIT Technical Report TR-CTIT-00-09, May 2000.

[4] D. R. H. Miller, T. Leek, and R. M. Schwartz. 'A Hidden Markov Model Information Retrieval System.' In the Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR-99), pp. 214-221, 1999.

[5] C. Monz. *From Document Retrieval to Question Answering.* ILLC dissertation series 2003-04, University of Amsterdam, 2003.

[6] W. A. Woods. 'Conceptual Indexing: A Better Way to Organize Knowledge,' Technical Report SMLI TR-97-61, Sun Microsystems Laboratories, Mountain View, CA, April 1997.