JHU/APL at TREC 2005: QA Retrieval and Robust Tracks

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

Abstract

The Johns Hopkins University Applied Physics Laboratory (JHU/APL) focused on the Robust and Question Answering Information Retrieval (QAIR) Tracks at the 2005 TREC conference. For the Robust Track, we attempted to use the difference in retrieval scores between the top retrieved and the 100th document to predict performance; the result was not competitive. For QAIR, we augmented each query with terms that appeared frequently in documents that contained answers to questions from previous question sets; the results showed modest gains from the technique.

HAIRCUT

The Hopkins Automated Information Retriever for Combing Unstructured Text (HAIRCUT) [3] is a document retrieval system developed at the Applied Physics Laboratory. It uses a traditional inverted index, a unigram language model for its similarity metric [2, 4], and a flexible tokenizer. The tokenizer supports words, stems, character n-grams, word n-grams and phrases. We have focused on language-independent techniques in developing HAIRCUT. It has been evaluated in TREC, CLEF and NTCIR in at least sixteen languages, and routinely performs among the top systems for both monolingual and translingual *ad hoc* retrieval.

Question Answering Information Retrieval Track

We were interested in whether we could identify terms that would be indicative of answers to particular types of questions. For example, it is reasonable to think that words such as *famous*, *himself* and *died* might be found frequently in or near answers to PERSON questions. If such terms can be accurately identified, they might be used to influence a document's ranking.

We used the TREC-2002, -2003, and -2004 questions and judgments to identify indicator terms. Our process begins by identifying a small taxonomy of question types, comprising HOW, HOW_MANY, WHAT_IS, WHEN, WHERE, WHO and OTHER questions. To automatically assign a question to its question type, we first parse the question using the Charniak Parser [1]. We then use a simple pattern-matching approach to map linearized parse trees onto question types. Any question that does not match a pattern is assigned to an 'OTHER' category.

Once the training questions are partitioned into question types, we want to use those assignments to identify indicator terms. We exploited the relevance judgments (qrels) from TRECs 2002 through 2004 for this purpose. For each question type, we identify two document sets: those that were listed as 'relevant' for questions of that type, and those that appeared in the judgments but that were judged not relevant. The QA judgments are not binary, but have four relevance values (descriptions taken from the track guidelines):

- incorrect: the answer-string does not contain a correct answer or the answer is not responsive;
- unsupported: the answer-string contains a correct answer but the document returned does not support that answer;
- non-exact: the answer-string contains a correct answer and the document supports that answer, but the string contains more than just the answer (or is missing bits of the answer);
- correct: the answer-string consists of exactly a correct answer and that answer is supported by the document returned.

For our purposes, we treat correct and non-exact as relevant, and all others as non-relevant.

Given these two document sets (judged relevant and judged non-relevant), we want to extract terms that are prominent in the relevant documents but not in the non-relevant documents. To do so, we first consider each document set separately, extracting words that appear frequently in that set, but relatively infrequently in the collection as a whole. We use a home-brew 'affinity' statistic for this purpose, but other measures, such as mutual information or the Dice coefficient, might work as well. The result is an ordered list of scored terms. We then score each term by the difference of its scores in the two sets. Finally, we select the top scoring terms as the expansion terms.

HOW	WHEN	WHERE
built	singer	war
park	death	north
scientists	thought	ago
orbit	America	rock
became	war	west
NASA	space	near
sun	history	museum
water	king	across
found	William	began
thought	II	miles

Table 1. Sample augmentation terms for three question categories

We generated augmentation terms for each question type. Sample augmentation terms are show in Table 1. Some augmentation terms are clearly sensible, such as *north*, *near* and *miles* for WHERE questions. Others, such as *rock* and *ago*, make less intuitive sense. Using retrieved but non-relevant documents to provide terms that should not appear in the augmentation lists eliminates most question-specific terms. However, there would certainly appear to be a fair amount of noise in these augmentation term lists. Nonetheless, we used the unaltered lists in our experiments.

To process each question, we first assign it a question type. We then build a new query, comprising the terms from the original query, plus the expansion terms for the selected question type. We weight query terms at a ratio of 100:1 relative to the expansion terms. This weighting was selected arbitrarily with no experimentation; a more accurate tuning of this parameter might lead to significant additional gains from the technique. Finally, we process the augmented query normally; we used HAIRCUT with words as indexing terms, a unigram language model with α=0.5, and no blind relevance feedback.

Our results showed a modest improvement from the use of this technique. Mean average precision without augmentation was 0.3322, while with augmentation it was 0.3417. The latter score ranked fifth among the submitted systems. Augmentation produced a 3% relative improvement. This gain was achieved with a simplistic question type assignment mechanism, and with no tuning of the weighting parameter. It is reasonable to expect that further gains might be seen if these two weaknesses were addressed.

Robust Track

In the Robust we looked for a simple predictor of retrieval effectiveness. We use a unigram language model for our similarity metric, and we were curious whether the document scores could tell us anything about performance. We therefore correlated various combinations of the scores of the top document, the tenth document, and the 100th document, with the average precision of the query, using the TREC-2004 Robust Track data as training data. We found the best correlations when taking the ratio of the score of the 100th document with the score of the top document. For 5-grams, the correlation was 0.57, while for words it was 0.53. We therefore used this ratio as our predictor for success on each query.

We were also interested in whether the use of phrases mined from the target collection could improve performance. We used suffix arrays [5] to identify all high frequency phrases in the document collection. We deleted leading and trailing closed class words from these phrases, and added the resulting phrase list as additional indexing terms.

We submitted five runs:

- **apl05pd**: description-only run using phrases.
- **apl05prf**: description-only run using phrases and blind relevance feedback.
- **apl05pt**: title-only run using phrases.
- **apl05cmb**: a combination of a word run, a character 5-gram run, apl05pd and apl05prf. We used z-score normalization to combine the runs, and weighted the four runs evenly.
- **apl05p50**: Under the theory that high-performing queries are more likely to benefit from blind relevance feedback, we selected each query's answer from either apl05pd or apl05prf according to whether it was predicted to be in the lower performing half or the upper performing half respectively. We used the ratio described above to predict into which half a query would fall.

Performance was uniformly poor relative to the field. Results are shown in Table 2. Based on these results, we cannot recommend the simple query difficulty metric we used in these studies.

Run	MAP	Geometric MAP
apl05pd	0.1411	0.0593
apl05prf	0.1654	0.0504
apl05pt	0.1374	0.0663
apl05cmb	0.1709	0.0890
apl05p50	0.1528	0.0607

Table 2. Performance on APL's five Robust Track runs.

Conclusions

Using ratios of language model retrieval scores to predict retrieval performance was not particularly effective relative to other techniques described at TREC-2005; we do not recommend its use. Assigning different document priors based on question type can produce a boost in retrieval performance. Our technique of adding query terms based on the question type is a way to exploit nonuniform document priors in any retrieval system without modifying the index. Our experiments were not finely tuned; our question type assignment was simplistic, and we tried only a single query term weight assignment. We nonetheless got a performance boost from the technique in the TREC-2005 QAIR task, as well as in our own cross-validation experiments on the question sets from the three prior TRECs. We therefore recommend the technique as a way to produce a modest boost in retrieval effectiveness for question answering systems. Careful tuning of the approach would likely increase that boost.

References

[1] Charniak, E. 'A maximum-entropy-inspired parser.' In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*. Seattle, Washington, pp. 132-139. 2000.

[2] Hiemstra, D., *Using Language Models for Information Retrieval,* Ph.D. Thesis, Center for Telematics and Information Technology, The Netherlands. 2000.

[3] Mayfield, J. and McNamee, P., 'The HAIRCUT information retrieval system.' *The Johns Hopkins APL Technical Digest* 26(1):2-14. 2005.

[4] Miller, D., Leek, T., and Schwartz, R., 'A hidden Markov model information retrieval system.' In *Proceedings of the 22nd Annual International ACM SIGIR Conference on R&D in Information Retrieval,* Berkeley, CA, pp. 214–221, 1999.

[5] Yamamoto, M. and Church, K. W. 'Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus.' In *Proceedings of the ACL Workshop on Very Large Corpora,* Montreal, pp. 28-37. 1998.