Combining Methods for the TREC 2003 Robust Track

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
{mayfield, mcnamee}@jhuapl.edu

Overview

The Johns Hopkins University Applied Physics Laboratory (JHU/APL) focused on the Robust Retrieval Track at this year's conference. In the past we have investigated the use of alternate methods for tokenization and applied character n-grams, with success, to tasks in ad hoc, filtering, and cross-language tracks.

For ranked retrieval, we have come to rely on a statistical language model to compute query/document similarity values. Hiemstra and de Vries describe such a linguistically motivated probabilistic model and explain how it relates to both the Boolean and vector space models [4]. The model has also been cast as a rudimentary Hidden Markov Model [7]. 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 $(1-\alpha)$ 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.

Earlier work on the Query Track, held during the TREC-8 and TREC-9 evaluations, showed that different query formulations could result in substantially different retrieval performance on individual queries (see Buckley [1]). For example, deleting an important query term, adding an important word that was not initially present, the use of idioms in topic statements, and deleterious effects caused by inappropriate stemming (over or under aggressive) were each shown to demonstrably alter precision on particular topics. Furthermore, multiple reports have appeared in the literature suggesting that a combination of evidence from multiple, disparate approaches can be beneficial (*e.g.,* Savoy [9]).

From this we conclude that a scheme based on multiple document representations and multiple similarity metrics might exhibit increased robustness in query performance, and possibly higher aggregate performance as well. How different methods can best be combined is not clear. In this paper we report on our efforts attempting to: (1) merge disparate run files; and (2) devise an automated technique for learning query-specific run weights that can be used to create a single, robust run.

We built several indices to compare different tokenization methods. We have begun investigating the use of part-of-speech tagging and entity-tagging to transform the term space, with the hope of capturing semantic distinctions (*e.g.,* bat, a noun, versus bat, the verb, or, Washington, the place, instead of Washington, a person); however, we did not use these indices in this year's Robust Track. Summary information for the indices that we used is shown in Table 1.

Table 1. Index statistics for the Robust Track collection.

		# terms	index size
words	w	554751	373 MB
words preserving case	c	698786	410 MB
stems (Snowball)	s	455803	320 MB
4-grams	4	251694	1.39 GB
5-grams	5	1485406	2.22 GB
6-grams	6	6030289	3.22 GB
words + phrases	p	19141479	1.97 GB

Disparate Retrieval Approaches

In the previous section we enumerated a number of alternatives to tokenization that resulted in different index data files being created for the collection. In many cases different tokenizations lead to similar overall performance. For example, on the query "health food" the use of case-normalized and unnormalized words should produce similar ranked lists. On the other hand, the query "NRA" will likely produce different results for case-sensitive words and character 4-grams.

We considered several approaches to computing document-query similarity:

- Retrieval without the use of blind relevance feedback
- Retrieval with blind relevance feedback using *t* expansion terms
- Massive collection enrichment
- Weighting query terms by setting qtf = 1
- RIDF weighting [2][6]
- Adjusting values for alpha, the parameter which regulates the relative importance of seeing query terms in documents in our statistical language model
- Applying UC Berkeley's logistic regression retrieval model [3]
- Calculating similarity with a probabilistic model and Okapi BM25 weighting [8]
- Requiring certain query terms and/or prohibiting others
- Expanding queries using a human-constructed thesaurus
- Query expansion using a statistical thesaurus

We were unable to consider each of these, primarily due to a lack of time to implement each method or to empirically evaluate each method with each index. We ended up using seven different indexes and 11 different retrieval methods (listed below). We then sought to combine or select from the 77 runs produced. With some risk of overtraining, we measured our performance on the 150 queries used in the TREC-6, TREC-7, and TREC-8 evaluations. We restricted ourselves to the use of only the 'Description' portion of topic statements; however, we did submit one official run using 'Title', 'Description', and 'Narrative' sections expecting that it would better contribute to the relevance pools.

For each of the 7 indexes, we created runs that computed document relevance by:

- [4 runs] Adjusting alpha values in our statistical language model of retrieval. We considered values of 0.2, 0.5, 0.8, and 0.9. We thought that higher alpha values would lead to better precision at low recall levels. Pseudo relevance feedback was not applied.
- [3 runs] Adjusting alpha values as mentioned above; however, relevance feedback was applied, selecting 60 'terms' (whatever those terms might be (*e.g.,* n-grams, words, stems), from 20 top-ranked documents. Alpha values of 0.2, 0.5, and 0.8 were considered. We imagine that it might also be useful to use other parameter settings for automated feedback, such as, different methods for isolating feedback terms, different numbers of expansion terms, or different numbers of presumed positive and negative documents.
- [1 run] We used our statistical language model with alpha = 0.5 without feedback, and *without stopword removal*. Normally we represent documents using all terms, but omit query terms at run-time that have a relative document frequency greater than 0.2.
- [1 run] Using the logistic regression retrieval model described by UC Berkeley
- [2 runs] Using Okapi BM25 term weighting in a binary independence model, both with and without relevance feedback (as described above). We used values of 1.2 for k1, 500 for k3, 0.6 for b, and we assumed that the top 8 documents for each query were relevant and all others were not, for the purpose of term weighting.

We considered several metrics for robust performance, but chiefly examined the percentage of topics with at least one relevant document in the top 10 ranks (TopTen) and the area under the curve when topical average precision is averaged over a number of the worst scoring topics and plotted as a function of the number of worst topics examined (MAP-Hardest). We focused on the hardest 25% of topics, as suggested in the track guidelines. For whatever number of topics is considered, MAP-Hardest is most effected by the most difficult topics. If the worst 12 topics are examined, then about three-quarters of the weight is given to the hardest 6 topics and only about one-quarter of the weight is given to the next 6 hardest topics. This puts a premium on doing as well as possible on the absolute hardest topics. We also considered high mean average precision (averaged over all topics) to be desirable, but were primarily concerned with 'robust' measures of performance.

We were interested to know how well each combination might do. In particular, we wondered how much improvement could be obtained given an oracle that could perfectly select the single-best method to apply for each individual query. Using the known relevance judgments for the TREC-6 through TREC-8 topics we determined that a method that selected the single best method (from our set of 77) for each topic could improve each of the performance measures appreciably. If this was not the case, for example, if our different methods did not exhibit large variations in retrieval performance, then any machine-learning approach to select a query-specific method would be doomed to failure.

Of our 77 runs, the one with the highest mean average precision (over all topics) on the training set used stems as indexing terms with the statistical language model (with alpha=0.2) and with relevance feedback. However, when alpha=0.5 mean average precision was about the same, and the robust measures were improved. We compare these runs and an oracular 'best' run using the robust metrics in Table 2.

Table 2. Comparing an oracle-based run and two high-performing methods on the training set.

	TopTen	MAP-Hardest	MAP
stems-lm2-rf	0.7467	0.0052	0.2513
stems-lm5-rf	0.8067	0.0061	0.2489
oracular	0.9600	0.0364	0.3587

From Table 2 we observe that an oracle-based run can improve mean average precision by approximately 40% and MAP-Hardest by roughly 600%.

Selecting a Single Method

We would like to predict which tokenization and scoring methods will prove most effective given a particular query. As several years of training data are available for this collection, we are inclined to adopt a supervised learning approach. For this study we applied Support Vector Machines (SVMs) [5]. For any learning approach to succeed we need to identify features that might discriminate high performing retrieval configurations from low performing ones.

We envisioned using a large set of features, including:

- Number of query terms
- Length of query in characters
- Length of query in words
- Capitalization pattern
- Digit pattern
- IDF of ith term
- Mean IDF
- Variance of IDF
- Whether ith term is a known closed-class word and which kind
- Whether words are stop-structure

In practice we used only four types of features: those based on the query alone; features based on various statistics of an index; features based on scores of documents in particular runs; and statistics computed from an index and from a run file.

- *Query-based*: total number of terms; number of unique terms; number of unknown terms
- *Using index*: maximum, minimum, mean, and variance of both query term IDF and RIDF; the number of documents containing: (1) all of the query terms; (2) all the query terms excluding stopwords; (3) the two rarest query terms; (4) the 2 most common query terms; and (5) the two query terms with highest mutual information.
- *Run-based*: the ratio of the score between the highest ranked and rank (5, 10, 20, 100, 500) documents using the 'stems-lm5-rf' run
- *Run plus index*: the percentage of unique terms to the total number of terms observed in documents from a specified range using the 'stems-lm5-rf' run (ranges considered included 0-10, 11-50, 51-100, and 201-300); the percentage of query terms to total terms observed in the ranked documents of a given range – as described above; and, the mean RIDF value of all terms occurring in the documents of the ranges described above.

Using these features we attempted to train a support vector machine with a cubic kernel for each of our 77 runs. For each of the queries in a 100-query training set we used the top 10 scoring runs as positive examples and the bottom scoring runs as negative examples. We hoped the SVM could distinguish methods likely to achieve high mean average precision (*i.e.,* good performance) and those unlikely to do so.

Unfortunately the SVM was not generally able to learn this distinction. The training algorithm converged, but essentially memorized the data. It may be that our set of features was inadequate and that more semantically laden features are essential to such a task. Or possibly, many more training exemplars are required for this approach to succeed. We next turned our attention to an approach based on combining results from multiple runs.

Merging Multiple Methods

As an alternative to selecting a single method based on features about the query, or weighting several methods predicted to perform well, we examined combination of multiple methods to produce a single ranked list for each topic. Combination of disparate techniques has occasionally led to improved performance in TREC-style evaluations; many consider that the constituent runs should be chosen to maximize orthogonality with respect to one another. That is, it is hoped that they make independent mistakes and that run combination will reinforce selection of good documents and lower the ranks of documents that only appear to be of high quality using a single method. In the past we have used combination to reasonable advantage; we found that combination of n-gram based runs with stem or word runs can confer a 10% relative advantage in mean average precision [6].

Our preferred method for merging multiple runs is to first normalize score values for each individual run and then create an ordering based on these normalized scores. We view scores as masses, and divide individual scores by the sum of the masses of the top *k* documents (we use *k=1000*). Because our probabilistic calculations are typically performed in log space, and scores are therefore negative, we achieve the desired effect by using the reciprocal of a document's score as its mass.

Other Approaches

We thought up several other schemes that were not implemented for the evaluation.

One was to build a tagger that processes query words and assigns them to different categories. For example, some words are clearly query-specific, such as "find documents that"; others are modifiers of a key concept, like 'international' in 'international organized crime"; still others are keywords, like 'black bears' in "black bear attacks". This technique is unproven, but recent successes in tagging applications might be applied to the 1000+ available topics from past TREC, CLEF, and NTCIR conferences. Terms in different categories could be treated (*e.g.,* weighted) differently.

Official Submissions

We submitted five runs described below.

aplrob03a was a combination of two runs, one using stems and one using character 5-grams. Title, Description, and Narrative topic fields were used here, but only the Description field was used for our other official runs. Relevance feedback was applied and alpha was 0.5. We thought that this method would maximize mean average precision over all topics. This run typifies our traditional processing.

aplrob03b was designed to maximize the number of topics with a relevant document appearing in the top ranks (say up to rank 20 or so). We filled 'slots' by examining several run files and selecting what we though to be good documents from different methods. Some of the documents were selected based on which run files worked well on the training set; others were based by clustering the top 50 ranked documents (for a given run) and selecting the highest ranking document from each of 5 clusters.

We used a quadratic implementation of agglomerative clustering that started with the 50 individual documents and repeatedly combined clusters, one at a time. Our distance metric was Sim(c1,c2) + Sim(c2,c1) – Max(cardinality(c1), cardinality(c2)). Our language model similarity metric is not symmetric, so we added the similarity between 'query' and 'document' (both clusters of documents) and the similarity between 'document' and 'query'. We then subtracted the cardinality of the larger of the two clusters; this was done in an attempt to even the distribution of the number of documents per cluster. We also ignored both very common and very rare terms when clustering. This method found a relevant document in the top 10 ranks for 134 of 150 topics in the training set.

To achieve reasonable performance in mean average precision as well, we then extended this list of top-ranked documents using run aplrob03d (described below), taking care to remove duplicate documents when building the ranked list.

aplrob03c was an attempt to maximize MAP-Hardest. We combined results using all 77 runs as input. On the training data this method performed the best, achieving 0.0103 on the MAP-Hardest metric. This is nearly double the performance obtained with a single, well-performing run, but well below our 0.0364 theoretical maximum.

aplrob03d is analogous to aplrob03a, but different in using only the 'Description' portion of the topic statements. Like aplrob03a, we expected this run to achieve good mean average precision over all topics and to serve as a baseline for our other runs.

Finally, ***aplrob03e*** was an overtraining run that sought to optimize TopTen. On the training data, we

were able to use the qrels to build a run by selecting the ith rank of the jth run for all topics. This method found a relevant document (in the top 10 ranks) for 143 out of the 150 training topics. Its use on novel data is questionable, but it is possible that the different runs selected represent a method for finding orthogonal methods. The following runs/ranks were employed:

stems-okapi	1
stems-logreg	3
stems-slm8	5
words-logreg	4
case-words-slm5	3
4-grams-okapi-rf	2
phrases-logreg	3
5-grams-slm8	5
stems-slm5	5
5-grams-okapi	2
4-grams-slm8-rf	5
words-nostop	5
4-grams-slm9	4

Examining Table 3 (below), we observe that run aplrob03c achieved a 0.0040 improvement in MAP-Hardest over our description-only baseline, aplrob03d; this is a 50% increase over the baseline. We interpret this as mild support for the hypothesis that combination of a very large number of methods can improve robustness. For the TopTen measure, run aplrob03e achieved a 90% success rate vs. 78% using our baseline method, a 15.4% relative improvement.

Table 3. Performance of officially submitted methods on all 100 topics. Run aplrob03d is a baseline for comparison against other methods.

	Fields	TopTen	MAP-Hardest	MAP
aplrob03a	TDN	0.8900	0.0238	0.2998
aplrob03b	D	0.8500	0.0113	0.2522
aplrob03c	D	0.8200	0.0120	0.2521
aplrob03d	D	0.7800	0.0080	0.2726
aplrob03e	D	0.9000	0.0096	0.2535

Conclusions

We attempted to determine whether selection or combination of diverse methods can improve the robustness of query processing. Our attempts to select preferred methods dynamically (*i.e.,* on a query-by-query basis) failed; however, we have not fully investigated this line of work. We did discover that combination of 77 runs (7 tokenizations and 11 similarity metrics) led to our best results for the MAP-Hardest measure. *A priori* selection of several diverse methods seemed to optimize the TopTen measure, though the small number of topics leaves it difficult to determine whether this result is valid. These results are based only on an examination of 'description-only' runs.

References

[1] C. Buckley, 'The TREC-9 Query Track'. Proceedings of the Ninth Text REtrieval Conference (TREC-9).

[2] Church, K. W., 'One term or two?' In Fox, E. A., *et al.*, eds., *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-95),* pp. 310-318. 1995.

[3] W. Cooper, A. Chen, and F. Gey, 'Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression', Proceedings of the Second Text REtrieval Conference (TREC-2), pp. 57-66.

[4] 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.

[5] T. Joachims, Making large-Scale SVM Learning Practical Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[6] P. McNamee and J. Mayfield, 'JHU/APL Experiments at CLEF-2001: Translation Resources and Score Normalization.' In: C Peters et al. eds., Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum (CLEF-2001), Darmstadt, Germany, pp. 193-208.

[7] 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, August 1999.

[8] S. E. Robertson, S. Walker, and M. Beaulieu, 'Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track'. In E. M. Voorhees and D. K. Harman (eds) Proceedings of the Seventh Text REtrieval Conference (TREC-7).

[9] J. Savoy, 'Cross-language information retrieval: experiments based on CLEF 2000 corpora.' In Information Processing and Management, Vol. 39(1), pp. 75-115, 2003.