Logistic Regression Merging of Amberfish and Lucene Multisearch Results

Christopher T. Fallen and Gregory B. Newby
Arctic Region Supercomputing Center
Fairbanks, AK
fallen@arsc.edu

November 15, 2005

Abstract

A simple logistic-regression based isolated data fusion algorithm was used to merge results from two free open-source text retrieval tools. The algorithm is described and results from each search tool are compared against the merged results and against each other. Basic performance measures are reported and discussed, and future projects are outlined.

1 Overview

The changing size and structure of the information available on the Internet guarantees that for any given performance measure, there will not be a single optimal method for retrieving information relevant to a specified topic. The availability and sophistication of distributed computing resources will continue to increase and consequently so will the relevance of problems related to distributed information retrieval. Data fusion and collection fusion are two such problems and the former in the context of a grid information retrieval meta-search application is the motivation behind the work described here.

Two GPL-licensed text retrieval tools, The Apache Software Foundation's Lucene and Etymon systems' Amberfish, were used to retrieve ranked document lists from the gov2-corpus document collection for each topic given in the efficiency, ad hoc, and named page tasks in the 2005 TREC Terabyte Track. For the ad hoc and named page tasks, ranked document lists were produced in three official runs: one run each using Lucene and Amberfish with automatically extracted queries, and one run that consisted of merging the results from the previous two runs using normalized linear least square fits to log-transformed relevance scores. For the ad hoc task an additional run consisting of merged Lucene and Amberfish results from manually extracted queries was also submitted.

Lucene

The Apache Software Foundation describes Lucene as "a high-performance, full-featured text search engine library written entirely in Java" [3]. Simple index and search applications were constructed by modifying the demo classes IndexHTML.java and SearchFiles.java, respectively, included with the 1.4.3 distribution of Apache Lucene.

Amberfish

Amberfish is a command-line based general-purpose text retrieval tool developed by Nassib Nassar and is described in greater detail at the Etymon Systems website [7]. Nassar used Amberfish to submit runs to the 2004 TREC Terabyte track and the results are summarized in his 2004 Terabyte track proceedings paper [8]. Amberfish 1.4.3 was used without modifications in the 2005 Terabyte track ad hoc task.

2 Prior Work

For the general fusion problem, Voorhees classifies merging strategies based on only the information available from a ranked document list as *isolated* and strategies that make use of collection information to merge document lists as *integrated* [10]. The logistic regression strategy described in section 4 is isolated and based on the document rank and relevance score information from a TREC-style ranked document list. The work described here is a preliminary step toward a study of the performance of new integrated merging strategies designed for Grid Information Retrieval (GIR) meta-search applications against a canonical set of isolated merging strategies similar to the study by Craswell, et al. [2]. The GIR architecture and requirements are described by the GIR working group[4].

The logistic regression approach to the search result merging problem is not new nor is it new to TREC. Savoy, Calvé, and Vrajitoru used logistic regression in the TREC-5 experiment as a strategy for both the collection fusion problem where search results from possibly disjoint and independent collections are merged into a single ranked list and the data fusion problem where engines operating on a single data collection interact to provide a single ranked list[9]. Estimating relevance is perhaps *the* fundamental challenge of information retrieval; a process to estimate the probability that a given document is relevant to a query from collection or document statistics is described by Gey as logistic inference[5].

3 Efficiency Task

The goal for the efficiency task was to demonstrate basic functionality of a simple JAVA IR tool built with the Lucene API. The index data structures were fed and saved to a remote NFS and a possible network driver bug caused intermittent slow-downs and system unresponsiveness. Consequently, only 66% of the document corpus was indexed before the

Measure	*Performance*	*Notes*
Time to index 280GB	80 hours	System slowdown from possible network driver bug
Index size	30GB	
Time to run 50000 queries	4.3 hours	0.33 seconds per query
Average precision after 20 docs retrieved	0.292	

Table 1: Efficiency task performance, Lucene

efficiency task deadline. The entire document corpus was indexed for the ad hoc and named page tasks.

Efficiency Task Results

The document text and index data structures were stored remotely on a NFS. The index data structure was built and searched with a commodity dual-Opteron Linux server. Listed below are two topics that threw a parse exception at search time and motivated the addition of simple exception handling to the demo Lucene search application:

Topic 35369 ?????????? ?????????? ????
Topic 46131 illinois school public relations ?ssociation

The median proportion of documents indexed is 100% but the Lucene index application indexed only 66% of the documents by the efficiency task deadline so a meaningful comparison of performance measures can only be made if the document collection is assumed to be homogeneous.

4 Ad Hoc Task

The ad hoc task was used to compare the performance of the Lucene and Amberfish IR tools and to evaluate a basic ranked-document list merging algorithm. Runs were submitted using both automatically extracted and manually constructed queries.

Lucene and Amberfish were used to create separate index data structures of the entire gov2 document corpus over a remote NFS. The corpus files were split into their respective document files before indexing the collection. Malformed HTML caused the Lucene index application to truncate about 1/50th of documents added to the index; the resulting index data structure was about 1/10th the size of the uncompressed document corpus. A corrupted file system prevented Amberfish from indexing one of the 273 document directories; the size of the index was 3/5th the size of the uncompressed document corpus. Both index data structures were partitioned arbitrarily for the purpose of load-balancing at search time, each

search application was run sequentially on four dual-Opteron nodes of a commodity Linux cluster. For each run, the queries were automatically constructed from the title fields of the 05.topics query file by joining the terms in each title with boolean-OR.

The results returned by each node were merged into a single ranked-document list based on document score and reported as runs ctfadhocaf1 and ctfadhocluc2, respectively. A variant of a logistic regression based database merging strategy [1] was used to merge the ranked-document lists from each search application. The logarithm of the document scores was calculated and linear least squares was used to fit log-normal curves to each of the resulting log score vs. rank data. The final merged document list was ordered by comparing the height of each curve at a given rank and picking the document corresponding to the largest score. This process was repeated for queries constructed manually and specifically for each search tool.

Ad Hoc Task Results

As measured by the mean average precision, Lucene performed somewhat better than Amberfish with both automatically extracted and manually constructed queries. As measured by binary preference, Lucene performed better than Amberfish with automatically extracted queries and Amberfish performed better than Lucene with manually constructed queries. By both measures the performance of the merged result list was comparable to the performance of Lucene alone. This is consistent with observations made at run time that the merge algorithm nearly always chose a document retrieved by Lucene over a document retrieved by Amberfish at each rank.

The motivation to fitting the relevance scores from each ranked document list to a log-normal distribution is to weight documents near the top of each list more heavily than documents further down so that presumably irrelevant documents far down the ranked document list will not pollute the presumably relevant results near the top of each list. While it appears from these results that the merge algorithm described above does not reduce the retrieval performance compared to either engine alone, the performance of the merge algorithm needs to be evaluated with respect to additional engines. Since the TREC Terabyte track ranked document lists contain all the information used by the merge algorithm, it is possible to use existing TREC Terabyte track data to further investigate merging performance.

Measuring by mean average precision and considering the topics where Lucene performed somewhat better than Amberfish using automatically extracted queries, the performance of the merged list was indistinguishable from the performance of Lucene except for topics 757, 789, and 797 where the merged performance was somewhat worse than that of Lucene alone and topic 758 where the merged performance was somewhat better than either engine alone. Amberfish somewhat outperformed Lucene in eight topics and of those topics, the merged performance was indistinguishable from Amberfish alone in topics 779 and 793. The merged performance of the merged list on the remaining six topics was somewhat worse than that of Amberfish alone.

Topic 758, perhaps by coincidence, is the one topic where the merged results perform better than the results from either engine alone is also a topic where both engines performed

Topic	*Query text*	*Amberfish*	*Lucene*	*Merged*	*TREC median*
751	Scrabble players	0.1009	0.0593	0.0590	0.2254
753	bullying prevention programs	0.1045	0.0400	0.0400	0.1869
757	murals	0.0394	0.1688	0.1345	0.2236
758	Embryonic stem cells	0.2308	0.2779	0.3084	0.6248
770	Kyrgyzstan-United States relations	0.0228	0.0018	0.0017	0.2683
779	Javelinas range and description	0.1019	0.0783	0.1011	0.3945
784	mersenne primes	0.5045	0.2890	0.3135	0.4560
786	Yew trees	0.1142	0.0829	0.0829	0.3616
789	abandoned mine reclamation	0.0242	0.0607	0.0515	0.2136
793	Bagpipe Bands	0.1019	0.0471	0.1014	0.2235
797	reintroduction of gray wolves	0.1252	0.4816	0.3125	0.5107
800	Ovarian Cancer Treatment	0.0583	0.0253	0.0270	0.1929

Table 2: Mean average precision for selected topics

well. The poor relative performance on topic 770 is likely because neither Lucene nor Amberfish parse "Kyrgystan-United States" effectively. There is no immediate explanation based on the query text for the inconsistent performance of the merged result list relative to the individual result lists from Amberfish and Lucene.

An ideal merge algorithm will combine all relevant documents from several ranked document lists into a single ranked document list. From these results, it is clear that the merge algorithm used here is far from ideal and more work is needed so that the merged performance is not capped at the performance of the best performing engine or to any engine in particular. There may not be a single good method for merging ranked document lists using only rank and relevance score information. The results from the TREC 2005 Terabyte track ad hoc task could possibly be used as a baseline from which to compare more sophisticated merge algorithms that take advantage of additional information like collection term distributions in the case of multiple disjoint document collections or term boost parameters of various engines operating on a single document collection.

5 Named Page Task

The goal of the named page task was to evaluate the basic performance of Lucene restricted to the collection of document titles. The index data structure from the ad hoc task was used and SearchFiles.java was modified so that each search was limited to the text contained within the title tags of each document.

	Auto query extraction			*Manual query construction*		
Measure	*Amberfish*	*Lucene*	*merged*	*Amberfish*	*Lucene*	*merged*
num_q	50	50	50	50	50	50
num_ret	50000	50000	50000	37996	44330	44405
num_rel	10407	10407	10407	9828	10407	10407
num_rel_ret	2205	3803	3825	2851	3584	3610
map	0.0499	0.0983	0.0969	0.0673	0.1099	0.1116
R-prec	0.1001	0.1709	0.1703	0.1283	0.1776	0.1800
bpref	0.1015	0.1349	0.1352	0.1371	0.1334	0.1336
recip_rank	0.4410	0.4725	0.4599	0.3915	0.4628	0.4853
ircl_prn.0.00	0.4785	0.5514	0.5416	0.4639	0.5517	0.5713
ircl_prn.0.10	0.1487	0.2761	0.2852	0.1982	0.3083	0.3031
ircl_prn.0.20	0.0968	0.1916	0.1937	0.1412	0.2105	0.2143
ircl_prn.0.30	0.0601	0.1236	0.1228	0.0816	0.1380	0.1402
ircl_prn.0.40	0.0399	0.0982	0.0891	0.0454	0.0814	0.0860
ircl_prn.0.50	0.0285	0.0647	0.0588	0.0290	0.0548	0.0581
ircl_prn.0.60	0.0059	0.0343	0.0305	0.0161	0.0419	0.0461
ircl_prn.0.70	0.0024	0.0143	0.0109	0.0082	0.0330	0.0352
ircl_prn.0.80	0.0002	0.0090	0.0062	0.0032	0.0255	0.0246
ircl_prn.0.90	0.0002	0.0005	0.0013	0.0032	0.0180	0.0173
ircl_prn.1.00	0.0002	0.0005	0.0013	0.0032	0.0038	0.0028
P5	0.2440	0.2960	0.2920	0.2920	0.3280	0.3280
P10	0.2080	0.2800	0.2820	0.2700	0.3200	0.3140
P15	0.1893	0.2813	0.2773	0.2533	0.3360	0.3307
P20	0.1880	0.2770	0.2740	0.2380	0.3300	0.3210
P30	0.1647	0.2693	0.2673	0.2193	0.3180	0.3140
P100	0.1126	0.2196	0.2200	0.1738	0.2540	0.2608
P200	0.0904	0.1705	0.1700	0.1298	0.1818	0.1858
P500	0.0618	0.1129	0.1132	0.0822	0.1108	0.1120
P1000	0.0441	0.0761	0.0765	0.0570	0.0717	0.0722

Table 3: Ad hoc task retrieval summary statistics

Measure	*Performance*	*Notes*
Amberfish		
Time to index 430GB	39 hours	No NFS slowdown observed
Index size	265GB	
Total time to return up to 1000 documents for each of 50 queries	4.2 hours	5 minutes per query
Average time to return top 20 documents	27 seconds	
Lucene		
Time to index 430GB	165 hours	NFS slowdown observed
Index size	43GB	
Total time to return up to 1000 documents for each of 50 queries	1.2 hours	1.5 minutes per query
Average time to return top 20 documents	2 seconds	
TREC efficiency task median values		
Time to index 430GB	16.5 hours	
Index size	63GB	
Total time to return up to 1000 documents for each of 50 queries	4.9 hours	5.9 minutes per query
Average time to return top 20 documents	0.47 seconds	

Table 4: Ad hoc task performance

Measure	*Performance*	*Notes*
Mean reciprocal rank	0.101	252 topics
Mean of median TREC reciprocal rank	0.3786	Stdev: 0.4240
Proportion of topics with the named page in the top 10%	21%	
Proportion of topics with no named page found	58%	
Proportion of topics with a reciprocal rank better than the median	8%	

Table 5: Named page task, Lucene

Named Page Task Results

The mean reciprocal rank of the named page was better than the median value in 21 out of 252 topics. Of those 21, the following topics scored a reciprocal rank greater than 0.33 with a median value less than 0.1:

Topic 632	rules for us federal government lodging and tax exemptions
Topic 658	genetics reference glossary
Topic 662	c++ allocator object documentation
Topic 720	1999 report romania human rights
Topic 834	walnut creek fishing conditions
Topic 838	denise crawford testimony

The StandardAnalyzer class in the Lucene API, used here to parse plain-text at index and search time, uses sophisticated grammar rules [6] to extract punctuation so it is possible that the strong relative performance for topic 662 is due to leaving the string "c++" intact. Relatively poor performance was observed for several topics that included the strings "us" for U.S. or numeric strings like "1999" to specify a year so there is no obviously plausible explanation for the strong performance on topics 632 and 720. Similarly, poor performance was observed for several topics containing double nouns as in "walnut creek" and "denise crawford."

6 Future Work

Merging results from different systems is a challenging but important contemporary problem. This paper has described an approach with fairly light requirements for system output, and a relatively simple logistic regression-based computational step to merge results. In order to be able to merge results across systems, we need effective means to determine how the individual "hits" in response sets should be ranked relative to each other. This is relevant

for Grid-based retrieval, for Web meta-search, and for other situations where a single query might be sent to multiple IR systems.

We can envision much more complicated approaches to that taken here, and will seek to compare these to the current logistic regression approach, as well as naive approaches (such as simplistic round-robin ranking, or ranking based on self-reported normalized cosine or percentage scores). There are three primary directions for more complicated approaches. First is to look to metadata about collections and response sets. We would like to consider how knowledge about relative collection size, the frequency of query terms in different collections, the number of hits for each query term, and other statistics about the collection and response set can be used to merge. From this, we envision modifications to traditional *tf*idf* equations that balance contributions from multiple sources, as a basis for re-ranking response sets.

Second is to look more deeply at response sets themselves. The downside of this approach is that different systems have different capabilities, and different ways of presenting results. In our multisearch experiments, we developed a simple XML envelope for results. More complete standards are under development, such as SRW (see `http://www.loc.gov/z3950/agency/zing/srw`), but might make even stronger demands of specific IR systems to present complete results. From a practical standpoint, we see that most systems are able to provide a normalized score for each document, an extracted plain text document title, and basic metadata (such as a URL or word count). Context-sensitive document extracts are sometimes available. We will look at how these factors can contribute to ranking, with an emphasis on simple equations such as logistic regression.

Third is to actually gather full text (or extracts) from documents linked in response sets. This is the approach taken by [2]. It has the benefit of using IR systems as "filters" for potentially interesting documents, then ranking them based on these documents' contents. The drawback is the heavier load placed on the contributing systems, to provide full text from each document in a response set.

Naive approaches will be useful for comparison. For example, would a simple divide-and-conquer approach using a single IR system, but with a subdivided collection, be as effective as a unified collection?

7 Conclusion

The work presented here has demonstrated that merging results across separate IR systems can be effective. We saw some TREC topics in the Terabyte ad hoc task where combined results were better than either system alone, and other results which were less effective than either system component. We consider result set merger a practical necessity for some types of collections, so will continue to investigate different approaches to merging. We will also continue development of a practical Grid-based design for collection management, indexing and query processing across data sets.

References

[1] Anne Le Calvé and Jacques Savoy. Database merging strategy based on logistic regression. *Information Processing & Management*, 36:341–359, 2000.

[2] Nick Craswell, David Hawking, and Paul Thistlewaite. Merging results from isolated search engines. In *Proceedings of the Tenth Australasian Database Conference*, 1999.

[3] Apache Software Foundation. `http://lucene.apache.org/java/docs/index.html`, 2005.

[4] K. Gamiel, G. Newby, and N. Nassar. Grid information retreival requirements. `http://www.gir-wg.org/`, 2003.

[5] F.C. Gey. Inferring probability of relevance using the method of logistic regression. In *Proceedings of the 17th International Conference of the ACM-SIGIR'94, Dublin, Ireland*, 1994.

[6] Erik Hatcher and Otis Gospodnetić. *Lucene in Action*. Manning Publications, 2004.

[7] Etymon Systems Inc. `http://www.etymon.com/tr.html`, 2005.

[8] Nassib Nassar. Amberfish at the trec 2004 terabyte track. In *The Thirteenth Text Retrieval Conference Proceedings (TREC 2004)*, 2004.

[9] Jacques Savoy, Anne Le Calvé, and Dana Vrajitoru. Report on the trec-5 experiment: Data fusion and collection fusion. In *The Fifth Text REtrieval Conference (TREC-5)*, 1996.

[10] Ellen M. Voorhees. Siemens trec-4 report: Further experiments with database merging. In *The Fourth Text REtreieval Conference (TREC-4)*, 1995.