Partitioning the Gov2 Corpus by Internet Domain Name: A Result-set Merging Experiment

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

November 15, 2006

Abstract

To study the MultiSearch problem and complete the Ad Hoc Task of the 2006 TREC Terabyte Track, the Gov2 collection was divided according to web domain and for each topic, the results from each domain were merged into single ranked list. The mean average precision scores of the results from two different merge algorithms applied to the domain-divided Gov2 collection and a randomized domain-divided collection are compared with a 2-way analysis of variance.

1 Introduction and Motivation

Result-set Merging

Modern IR tools need to operate on document collections of enormous size and scope. The size of the collection and the distributed nature of the information itself often requires a divide and conquer approach where the collection is divided into smaller sub-collections and IR operations like keyword search are applied to the smaller sub-collections individually before merging into a single ranked list.

The number of possible ways to divide a collection of any reasonable size is always astronomical. Sometimes the divisions are arbitrary with respect to the content of the documents for purposes of load balancing. Or the divisions reflect the network topology of the machines that store the data. One way to divide the set of web pages found in nature is by web domain. If a distributed IR application were used to search each web domain individually, how should the results from each domain be weighted so that the results can be merged and presented to the end user or `trec_eval`?

The purpose of this study is to build an experiment using the TREC Terabyte Track topics and the Gov2 corpus to determine the relative IR performance of two result set merging techniques in a simulated Internet-scale Grid Information Retrieval (GIR) environment. The model used to design the experiment is of a hypothetical GIR application that searches each of the approximately 17,000 unique web domains in the Gov2 corpus individually as if using a local search application provided by each domain and then merges the result sets into a single ranked list to be presented to the user or to be evaluated with `trec_eval`. The web domains in the Gov2 corpus range in size from one to nearly one million pages so any merge method in this experimental framework needs to merge thousands of result sets with a similarly wide range of lengths.

Grid Information Retrieval (GIR)

One of the main inspirations for this work is our continued interest in distributed information retrieval systems. The author and colleagues are involved with the Open Grid Forum's [12] standards working group on Grid Information Retrieval [6], seeking to produce standard methods for IR systems to interpolate.

GIR spans several major themes: distributed indexing, transport methods for queries and result sets, human interface, and methods for query persistence. For TREC purposes, though, the emphasis is on result-sets merging. The issue is that different IR systems, with different collections or subcollections, each produce their own results for a given query in a distributed IR system. How can these different sets of results (each ordered by relevance, as produced by the independent IR systems), be effectively unified into one set?

Rather than ranking arbitrarily, the goal is to produce a single relevance-ranked set, with ordering that indicates the relative relevance of each document, regardless of which IR system it came from. In this paper, we present our exploration into doing this relevance re-ranking.

2 Experiment Overview: Apparatus

Machines

A standalone server and a Cray XD1 system were employed this year for indexing and searching. An additional pair of servers was used for testing and development of the MultiSearch capability described in the previous section.

For development and initial testing, a dual Xeon 64-bit system was used. This system, from ASL Inc., features 8GB of memory and 3.6TB of disk space, running Linux. The large disk space allowed for several different iterations of the indexes to be built, during early deployment of the experimental methodology described below. This system built many of the indexes, which were later copied to the larger system.

The larger system is a Cray XD1, "nelchina." Nelchina features 108 2.6Ghz Opteron processor cores with 4GB of memory each, the PBS Pro scheduler, and Cray's variation on the SuSE Linux operating system. A disk subsystem, provided by Direct Data Networks, provides 18TB of high performance disk space for temporary storage. By experimentation, we found that about six simultaneous indexing or searching threads could operate simultaneously, before the aggregate performance would decrease (keep in mind that each searching thread would need to open about 17000 separate Amberfish indexes).

Amberfish

Amberfish is a free command-line based general-purpose text retrieval tool developed by Nassib Nassar and is described at the Etymon Systems website [11]. The performance of Amberfish in the TREC Terabyte is described by Nassar in the 2004 TREC proceedings [10] and by Fallen and Newby in the 2005 TREC proceedings [4]. Version 1.5.10 of Amberfish was used without modifications in this 2006 TREC Terabyte Track experiment.

Corpus Partitions

A *partition* of the set of documents in the Gov2 corpus is a collection of disjoint subsets that contains the original set[9]. Two different partitions were constructed at index time, the *domain partition* and the *randomized domain partition*, described below. Results from the domain partition were submitted as official TREC runs and results from the randomized domain partition are used for statistical comparison in order to quantify the effects of partitioning a collection in different ways on the performance of the result-set merging algorithm.

Let G be the set of documents in the Gov2 corpus and let $\mathcal{G}$ be a partition of G. Then the elements S_i of $\mathcal{G}$ are subsets of G such that $S_i \cap S_j = \emptyset$ for each i and j so the subsets are pairwise disjoint and $G \subseteq \bigcup_i S_i$ so the subsets cover G. The cardinality or size of a set S is defined as the number $|S|$ of elements or documents in the set. Note that this formulation of the distributed IR problem excludes the common situation when multiple IR systems crawl a single large dataset and likely index many of the same documents.

The subsets or subcollections of the domain partition $\mathcal{G}_d$ are sets of pages with a common domain name. That is, if d_i and d_j are two documents in the subset S that is itself an element of $\mathcal{G}_d$, then the d_i and d_j are documents with identical domain names in their URLs. One goal of the experiment is to try and isolate the

IR measurable effects of grouping related pages together at index-time — where d_i is "related" to d_j in this context means d_i is "in the same domain as" d_j — by comparing results retrieved from collection indexes defined by $\mathcal{G}_d$ and indexes defined by a partition $\mathcal{G}_r$ with the same subset size distribution as $\mathcal{G}_d$ but with pages assigned to each subset at random. The partition $\mathcal{G}_r$ is defined as follows.

From the first web domain subset S_1 in $\mathcal{G}_d$, form a subset R_1 of $|S_1|$ unique pages drawn at random from G. Form the second subset R_2 with $|S_2|$ pages drawn from G that are not already in R_1. In this manner construct the partition $\mathcal{G}_r$ inductively by filling R_i with $|S_i|$ pages such that

$$R_i \subseteq G - \bigcup_{j<i} R_j. \tag{1}$$

There are very good, practical reasons why this procedure of dividing the Gov2 collection by domains before indexing is not often used; one is that *there are very many domains on the Internet and a few of them are very big*. Consequently, search time on each of the large domains and over the many small domains are affected significantly for the worse. The second problem with this partitioning scheme is closely related to one of the fundamental problems of result set merging across IR systems: weighting results from one collection against results from another collection without downloading the documents themselves or using a priori characterizations of each collection searched is not straightforward. Dividing an Internet-scale collection by domains amplifies this second problem significantly so in a sense, partitioning by domain is an instructive model to explore problems in distributed information retrieval.

Domains in the Gov2 Corpus

The Gov2 corpus contains approximately $17,000$ unique domains, tens of domains contain multiples of $100,000$ TREC IDs each; thousands of domains contain less than 10 pages each; and in Figure 1 a rank-size plot illustrates that the size distribution between these two extremes looks to be linear on a log-log scale implying a power-law fit to the rank-size curve. Both of the partitions $\mathcal{G}_d$ and $\mathcal{G}_r$ used to construct indexes have size distributions illustrated in the figure. Power-law distributions found in collections of documents from the web are not new or surprising. The distribution of the number of links pointing to or from each document in a collection is known to be a power-law [1][5] as is the distribution of domain sizes on the Internet [8]. In this paper "size distribution" is used colloquially to refer to the rank-size ordered distribution whose graph looks similar to but represents different information than a power-law probability distribution[7].

A practical description of the domain size distribution is that there are a few domains in Gov2 with an enormous number of pages and there are an enormous number of domains with an effectively infinitesimal number of pages. Another important characteristic of the domain-size distribution is that the variance of the elements of a power-law distribution is unbounded so that the descriptive statistic "mean domain size" may not be very descriptive at all in the sense that as domains are added to a sample of a collection, the mean domain size will exhibit a random walk: each large domain is large enough to significantly increase the mean domain size no matter the size of the sample. Similarly, there are enough small domains added to the sample between the large domains to significantly decrease the mean.

The names of the largest and smallest domains, along with a selection of mid-sized domains are listed in Table 1. Note that many of the large domains are library sites or repositories of specialized information, the smallest domains are little more than placeholder or temporary pages, and the medium-sized sites are more varied. Therefore it may be useful to craft unique GIR search, filter, and merge algorithms to specific ranges of domain sizes.

Experiment Method and Result-set Merge Algorithms

The Gov2 corpus was split into individual document files and stripped of the TREC headers. To index the domain partition, a separate file constructed from the TREC headers was used to associate TREC IDs to domain names. For each domain, a list was created containing the TREC IDs contained in the domain. Using

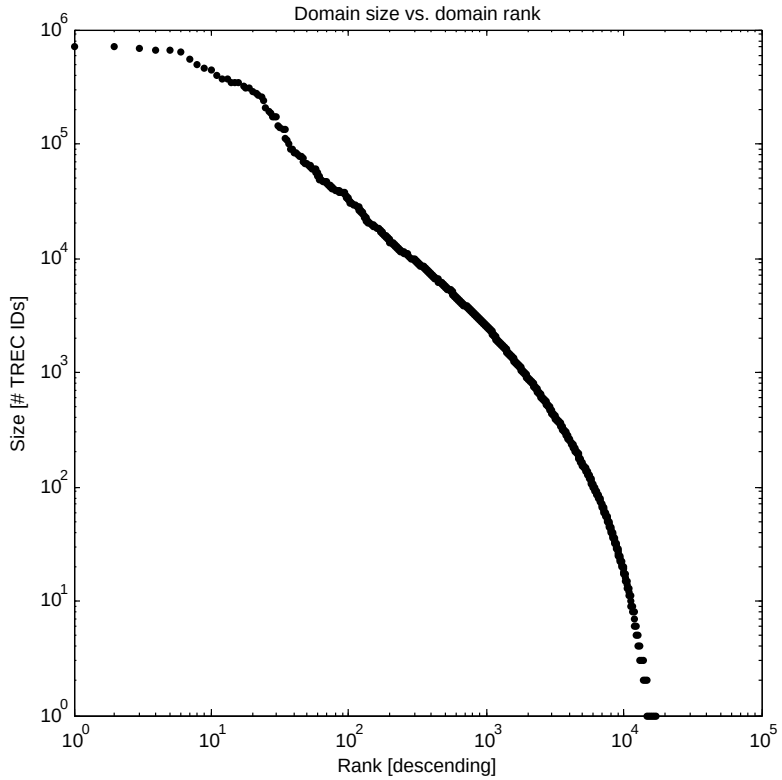

Figure 1: *Sizes of web domains in the gov2 corpus as measured by TREC ID count*

Large		**Medium**		**Small**	
717,321	ghr.nlm.nih.gov	10,630	www.newton.dep.anl.gov	1	adm.state.ky.us
709,105	nih-library.nih.gov	10,595	www.hq.nasa.gov	1	admin.monroecounty.gov
694,505	wcca.wicourts.gov	10,514	www.oalj.dol.gov	1	ad02dwvchr.er.usgs.gov
665,987	cdaw.gsfc.nasa.gov	10,486	fl-cvs.mcs.anl.gov	1	acousticaltest.grc.nasa.gov
650,208	catalog.tempe.gov	10,445	www.nga.gov	1	acmb.larc.nasa.gov
637,313	www.catalog.kpl.gov	10,418	www.atsdr.cdc.gov	1	aces.calpers.ca.gov
551,123	edc.usgs.gov	10,348	www.glerl.noaa.gov	1	acb700.tc.faa.gov
492,416	www.fs.usda.gov	10,223	www.sanantonio.gov	1	abcprod.mstc.state.ms.us
459,329	gis.ca.gov	10,218	www.pnl.gov	1	aasis.state.ar.us
441,201	www.csm.ornl.gov	10,189	aether.lbl.gov	1	aaa.lanl.gov
403,648	www.fgdc.gov	10,187	nepp.nasa.gov	1	a841-dotweb01.nyc.gov
367,371	www.archives.gov	10,066	www.fsa.usda.gov	1	a069-webapps3.nyc.gov
363,942	www-oss.fnal.gov	10,064	www.epa.state.oh.us	1	2002.cancer.gov
342,746	www.census.gov	10,030	www.dhs.state.tx.us	1	2001.nci.nih.gov
340,608	www.ssa.gov	10,016	www-pao.ksc.nasa.gov	1	100.webstaging2.cit.nih.gov

Table 1: *A selection of domain names and associated TREC ID counts*

the command-line interface to Amberfish, a single inverted index for the documents in each list (domain) was created from an "index worker" script that indexed each domain in serial by selecting domain lists from a queue of unindexed domains.

The randomized domain partition was indexed by the same scripts used on the domain partition after each randomized domain list was constructed by counting the number of documents in each domain list and drawing that many pages without replacement and at random from the master list of TREC IDs. The filenames of each domain list and associated Amberfish index file contained a number that allowed the actual domain name to be referenced in later analysis; the filenames of each randomized domain list and index were identified with the same number as the corresponding domain list. As will be described below in the discussion of results, the choice of index file filenames potentially biased the results by weighting domains whose filenames appear at the top of a GNU `ls` listing more than those domains whose filenames appear later.

At search time, the list of topics was divided and the parts were assigned to "search worker" scripts that, for each topic, called the Amberfish search program on every index corresponding to a domain in a partition. The results from each topic and index (domain) pair were saved to a text file in TREC format and named according to the topic and index name. When the search worker script had searched every index in the partition, the results were for a topic were merged into a single ranked list using merge algorithms described below. By saving the results for each topic and domain pair in separate files, the performance of different new merge algorithms can be compared without searching the collection anew.

The *sort-merge* algorithm used as a comparison baseline is the GNU `sort` command applied to the document score column of all the TREC result files in a partition for a given topic. The *log-merge* algorithm is the same algorithm used by the authors in TREC 2005 Terabyte track [4] and is inspired by the logistic regression algorithm described by Savoy, Calvé, and Vrajitoru[13][3]. In their papers, the score vs. log(rank) data for each result set to be merged is first fit to a logistic curve using the method of maximum likelihood to estimate the parameters of the best-fit logistic function and then each document in a result-set is merged into a single ranked list according to its height on the logistic curve evaluated at the log of its document rank in the original result-set. In the log-merge algorithm used here, linear least squares is applied to a modified logistic function to estimate the parameters of the logistic function.

Let $\{(x_i, y_i, d_i)\}$ be a result set for a topic from a single domain. The i^{th} ranked document is d_i, $x_i = \log i$, and y_i is a monotonically decreasing sequence of relevance scores. The task is to assign parameters l and m such that the logistic function

$$f_{l,m}(x) = \frac{e^{l+mx}}{1 + e^{l+mx}} \tag{2}$$

best fits the result set in the least squares sense of minimizing the sum of the squares of the residuals. A simpler problem is solved instead by taking the logarithm of the equation above

$$(\log \circ f)_{l,m}(x) = l + mx - \log(1 + e^{l+mx}) \tag{3}$$

to get an expression that very nearly describes a line, at least when applied to the IR data sets of interest. Then a straightforward application of a linear least squares routine yields estimates for the intercept and slope of the best-fit line. The value of the parameter m can be set to the estimated slope directly and the parameter l can be found by setting $x = 0$ and solving the equation $l - \log(1 + e^l)$ set to the estimated intercept. A new set of log-normalized scores for each result-set is calculated from $f_{l,m}$ and the result-sets are merged in descending order with respect to the new log-normalized scores.

3 Experiment Results: The Ad Hoc Task

System Performance

The Cray XD1 at the Arctic Region Supercomputing Center is primarily used as a shared resource where individual processing nodes are allocated through the PBS Pro batch scheduler. Corpus data and index

Machine	Index wall time	Throughput	Index size	Partition
Cray XD1	470 [hr]	0.9 [GB/hr]	264 [GB]	Domain (Ad Hoc Task)
ASL Box	800 [hr]	0.5 [GB/hr]	264 [GB]	Randomized domain (comparison)

Table 2: *Amberfish index performance*

data structures used for the TREC Terabyte Track experiment resided on the Lustre Cluster File System (CFS) so the I/O bandwidth available at the time of the experiment depends on the number and type of jobs submitted by multiple users and as such it is difficult to accurately measure TREC Terabyte Track system performance on the XD1 without dedicating the entire machine. Both index and search operations were submitted in stages through batch jobs and the number of jobs running in parallel may also affect the the performance of each job individually. In addition, the index process was interrupted by system downtime. Consequently, the index and search times reported for the Cray XD1 are only rough estimates and no attempt was made to either optimize or carefully assess system performance. Regardless, the times required to index 426 GB reported here are rather large and a significant cause of this is the one-to-one correspondence of domains to Amberfish index files: some domains are very large and many domains are very small.

Each node of the Cray XD1 used in this study contains two 64-bit AMD Opteron 250 processors that share 8 GB of RAM [2]. Two nodes with three to four concurrent Amberfish processes per node were used to index the domain partition of the Gov2 collection used in the Ad Hoc task. Each "index worker" processes selected domains to index from a queue. For informal performance comparison, a generic dual Xeon from ASL with two processors at 3.8 Ghz and hyperthreading enabled, and 8 GB of RAM was used to index the randomized domain partition using three to four concurrent Amberfish processes as on the Cray XD1. The ASL system has a RAID5 array with approximately 3.6TB of space in 13 drives, using SATA-II drives and controllers. Downtime and unknown system loads from other users will also affect the accuracy of the index times reported for the ASL machine.

For the automatic search run, a total of four concurrent Amberfish "search workers" were started on two dual processor nodes on the XD1. The list of topics was divided among the search workers and each worker processed the topics sequentially. For each topic, the search worker ran the Amberfish command on each domain individually and the top 1000 results from each domain were saved to a file in official TREC result format. After a search worker recorded results from every domain for a given topic, the result files were merged into a single ranked list using the GNU `sort` command applied to the relevance score and then merged again to another ranked list using the logistic normalization procedure described above. By saving the results for a topic from each domain individually, merge algorithms based on the information contained in the official TREC format result-sets and domain-specific meta information can be tested and compared using relatively modest hardware without searching the collection again.

This flexibility with respect to result-set merging experiments comes at the price of certain end-user features. For instance, restricting search results for a topic to the top 20 hits will not significantly improve performance with this system because the process of opening each of the 17,000 index files corresponding to the web domains contributes significantly to the total search time. Some performance improvement may be noticed if only the top 20 results from each domain are saved because some of the domains are very large, but this will also affect the IR performance of the log-merge algorithm because the relevance scores from every result in a result set are used in the renormalization procedure. Consequently, the official search times reported are the times to return up to 10,000 documents per topic.

For the manual search run, rather complex queries were constructed from the topic descriptions with Boolean AND, OR, and NOT. Hence, the manual queries contained more terms on average than the automatically constructed queries and this impacted search performance over the larger domains. In order to complete the manual run before the submission deadline, four dual processor nodes were used instead of the two used in the automatic run but otherwise search and merge procedure was the same as described above.

Run Tag	Total search time Wall	CPU	Throughput	Merge method
149 queries extracted automatically by joining topic title terms with AND				
arscDomAlog	2,000 [min]	8,000 [min]	13 [min/topic]	Log
arscDomAsrt	2,000 [min]	8,000 [min]	13 [min/topic]	Sort
50 queries constructed manually from topics 801-850				
arscDomManL	2,500 [min]	20,000 [min]	50 [min/topic]	Log
arscDomManS	2,500 [min]	20,000 [min]	50 [min/topic]	Sort

Table 3: *Amberfish Ad Hoc Task search performance on the Cray XD1 and the domain partition.*

IR Performance

Similar to the published results in the TREC 2005 Terabyte track [4], the `trec_eval` IR measures of the results returned by Amberfish are well below the TREC participant median which is to be expected as Amberfish does not use keyword expansion or other sophisticated techniques beyond Boolean set operations to retrieve results. Searching across a very large number of indexes split according to web domain is not likely to help the results either as information like relative frequencies of search terms in two documents are not going to available if the two documents are from different domains. However, the goal of this study is to investigate the effectiveness of result-set merge algorithms as applied to a large distributed information retrieval system.

In both the automatic query extraction run on topics 701-850 and the manual query construction run on topics 801-850, the mean values of the mean average precision (MAP) calculated by version 8.1 of `trec_eval`, recorded in Table 4, for the log-merged results are larger than the corresponding MAP for the sort-merged results. Note that the standard deviation of the MAP in the automatic runs is greater than the mean value of the MAP itself. This slight performance improvement of the log-merged results over the sort-merged results on a web-domain partitioned collection is consistent with the results observed in the TREC 2005 when the collection was divided arbitrarily into a small number of subcollections with uniform size.

To quantify the effects on IR performance due to the merge methods used as well as the effects due to eliminating the natural corpus structure defined by web domains by dividing the corpus arbitrarily with respect to the document content at index-time, the mean values of the MAP taken over the merged result-sets from 149 automatically extracted queries applied to the domain partition and the randomized domain partition are recorded in Table 5. As observed in the official TREC results from 2005 and 2006, the log-merge method outperforms the sort-merge method regardless of whether the underlying collection is partitioned by web domain or partitioned by randomized web domains. The MAP scores from the log-merge and sort-merge methods applied to the randomized domain partition are slightly better than the respective MAP scores for the domain partition but this difference is not likely to be significant as determined by the results of the 2-way analysis of variance reported in Table 6. Note that the mean and median MAP values reported in Table 5 imply that the normality assumption in the ANOVA test is not satisfied by this data.

4 Discussion

From the analysis of variance referenced above, it is clear that randomly distributing pages among a fixed domain size distribution does not significantly affect search performance. Nonetheless, when the number of domains that returned results for each topic is plotted as in Figure 2, the number of "domain hits" per topic is consistently larger for the randomized partition than for the domain partition. This suggests that the retrieved results, at least those found by Amberfish, are concentrated in a smaller number of Internet domains than would be expected if the results were distributed randomly over the domains. If *relevant* retrieved results also exhibit a similar distribution, then identifying those domains that contain a larger

	Mean	Standard Deviation
149 queries extracted automatically from topics 701-850		
arscDomAlog	0.0571	0.0801
arscDomAsrt	0.0468	0.0676
TREC Participant median	0.2922	0.1725
50 queries constructed manually from topics 801-850		
arscDomManL	0.0351	
arscDomManS	0.0278	
TREC Participant median	0.2436	0.1464

Table 4: *Official 2006 TREC Terabyte Track Results: Average Precision scores averaged over the topics indicated*

		Average Precision			Total Relevant
Partition	**Merge method**	**Mean**	**Median**	**Stdv.**	**Documents Returned**
Domain	Log	0.0571	0.0240	0.0801	15,484
Domain	Sort	0.0468	0.0201	0.0676	14,987
Randomized	Log	0.0624	0.0304	0.0790	15,112
Randomized	Sort	0.0485	0.0209	0.0639	14,853

Table 5: *Mean Average Precision and number of relevant documents retrieved from the 26,917 document TREC relevant document pool. Statistics calculated over 149 topics with automatically extracted queries.*

	SS	MS	p
Columns	0.0216	0.0216	0.0445
Rows	0.0019	0.0019	0.5508
Interaction	0.0005	0.0005	0.7609
Error	3.1561	0.0053	
Total	3.1801		

Table 6: *Results from a 2-way ANOVA test with 149 repetitions (topics) on MAP scores with merge method represented by columns and partition scheme represented by rows.*

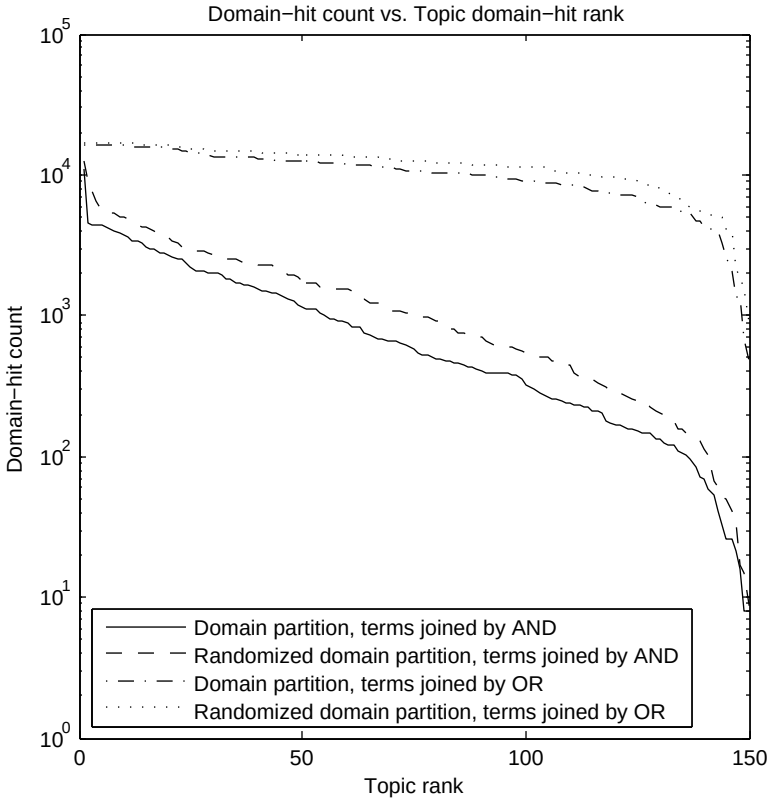

Figure 2: *Size vs. rank plot of the number of domains returning hits for queries constructed automatically from TREC Terabyte Track topic titles*

concentration of relevant results for some training set of queries and weighting the results from those quality domains accordingly may be an effective strategy to improve result-set merging performance.

Seemingly little information is available from the relevance score vs. rank profile used by the log-merge method. Nonetheless, the log-merge method does significantly improve result-set merging performance relative to a straightforward sort operation on relevance scores. However, observations made during the experiment suggest that the log-merge method is not particularly well suited to a domain-partitioned collection or the relevance scoring algorithm used by Amberfish. For instance, many domain result-sets contained very few results; many domains returned a single result per query. And since Amberfish does not return cosine-like relevance scores but instead scores the top ranked document in a result-set at $10,000$ regardless of the actual similarity of the top-ranked document to the query, there is little variability among log-normalized scores across domains returning few hits because very few points are used to fit the logistic curve. So for the worst case of merging across many result-sets containing a single result, "relevance" is assigned according to the order in which the result-sets are merged because every result has the same relevance score. In this experiment each result-set was saved to its own file before merging so the merge order was effectively determined by `ls`.

5 Conclusions and Future Work

Partitioning a large Terabyte-scale collection of web pages by Internet domain name is not a good design for a monolithic IR application for both machine performance and IR performance considerations. However, the web itself is naturally partitioned by Internet domain name so a distributed GIR application may need to deal with the complications described above anyway. In particular, methods for assigning relative relevance scores to results from different collections that are robust with respect to the number of results in a result-set are needed. If the IR application used returns cosine-like relevance scores then the log-merge method may in fact be robust in that sense. Given the wide range of domain sizes, perhaps different merge methods designed for particular ranges of domain sizes may be more effective than the single merge method approach used here. An in-depth study of the concentrations of the relevant results in each domain for a set of typical queries may also yield additional meta-information useful for result-set merging.

References

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world-wide web. *Nature*, 401:130, 1999.

[2] Introduction to using the Cray XD1 at ARSC. `http://www.arsc.edu/support/howtos/usingxd1.html`, 2006.

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

[4] Christopher T. Fallen and Gregory B. Newby. Logistic regression merging of amberfish and lucene multisearch results. In *The Fourteenth Text REtrieval Conference*, 2005.

[5] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication ACM SIGCOMM'99*, pages 251–262, 1999.

[6] The GIR Working Group. `http://www.gir-wg.org/`, 2006.

[7] R. Günther, L. Levitin, B. Schapiro, and P. Wagner. Zipf's law and the effect of ranking on probability distributions. *International Journal of Theoretical Physics*, 35:395–417, 1996.

[8] Bernardo A. Huberman and Lada A. Adamic. Growth dynamics of the World-Wide Web. *Nature*, 401:131, 1999.

[9] James R. Munkres. *Topology*. Prentice Hall, 2000.

[10] Nassib Nassar. Amberfish at the trec 2004 terabyte track. In *The Thirteenth Text REtrieval Conference Proceedings*, 2004.

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

[12] The Open Grid Forum. `http://www.ogf.org`, 2006.

[13] 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*, 1996.