IIT at TREC 2004

Standard Retrieval Models over Partitioned Indices for the Terabyte Track

Jefferson Heard
Information Retrieval Laboratory
Illinois Institute of Technology
Chicago, Illinois
heard@ir.iit.edu

Ophir Frieder
Information Retrieval Laboratory
Illinois Institute of Technology
Chicago, Illinois
ophir@ir.iit.edu

David Grossman
Information Retrieval Laboratory
Illinois Institute of Technology
Chicago, Illinois
dagr@ir.iit.edu

ABSTRACT

For TREC-2004, we participated in the Terabyte track. We focused on partitioning the data in the GOV2 collection across a homogeneous cluster of machines and indexing and querying the collection in a distributed fashion using different standard retrieval models on a single system, such as the Robertson BM25 probabilistic measure and a vector space measure. Our partitioned indices were each independent of each other, with independent collection statistics and lexicons. We combined the results as if all indices were the same, however, not weighing any one result set more or less than another.

1. INTRODUCTION

For the TREC 2004 Terabyte track, we focused on indexing and querying the data with our existing AIRE engine [3]. We partitioned the 480GB GOV2 data across fourteen machines and built independent indices for each segment. We then analyzed the statistics for each of these independent indices to make sure that the collections were roughly equal in size and characteristics. After satisfying ourselves that this was the case, we issued each query on each system and combined the results to get the required number of documents for each query.

2. EXPERIMENTAL SETUP

2.1 Indexing

The hardware for our tests consisted of fourteen machines networked on a dedicated switch. Each machine was equipped with 2 10,000 RPM 78 GB SCSI drives, 2 AMD K7 Athlon MP 2400+ processors, 2GB of RAM, and RedHat Linux Advanced Server 2.1. The indexing of the system was done using the BEA JRockit JVM, while the querying was done using the Sun JDK 1.4.2_04. The system for this set of experiments was very much like last year's, with no changes introduced in order to be able to index or query the large amount of data needed for the Terabyte track.

The indexing engine was set up in as simple a way as possible to ensure maximum indexing speed and that it would run to completion. No stemming was done, nor was any link information or entity extraction mechanism used. Phrasing was used, with a limit of two words per phrase and a minimum of 25 occurrences of a phrase to be entered in the lexicon.

The data was partitioned out across each machine roughly equally, and left compressed in gzip format. No directory in the collection was split out across two separate machines, however. Documents were chosen for each machine by picking directories of the GOV2 collection sequentially as listed by ls and trying to fit them together so that each machine had roughly 34GB of data.

2.2 Querying

The query engine was set up to do no stemming nor use any link information. We ran two query runs: one with the IIT vector-space similarity function [3], and one with the Robertson BM25 probabilistic similarity function [4]. Only the query title was included in the query.

Since all queries were issued to each machine, we needed a way to combine the the independent results to come up with one master result set. Since the sets of collection statistics were similar, we decided to treat each result set equally for each query and combine them by simply concatenating the results and sorting them based on non-normalized relevance score, as in the section on "perfect merge" in [1]. Not normalizing the score was important, since normalizing would have caused the top documents from each index to be treated equally by virtue of having the highest score for the index rather than the highest score overall.

3. RESULTS

3.1 Indexing

Indexing the entire GOV2 collection took 8.2 hours from the first machine to start until the last machine to end. Each index was approximately 4GB in total. The lexicon took up an average of 260 MB, while the posting lists took an average of 3.05GB. The document data averaged 600 MB. Each lexicon had between 5.8 and 6.9 million words, between 1.4 and 1.8 million phrases. Standard deviation was 5% of the mean number of words for all lexica on all machines, and 7% for phrases. For document count and posting entry count, the deviation was a mere 2% of the mean number of documents. The average number of distinct terms per document in the collection (see Table 2) was similar across the cluster, with a meager standard deviation of 7 versus a mean of 221.53 distinct terms per document. Similarly, the average total number of terms per document varies only slightly across collections.

The only statistics that really vary widely are the maxi-

Table 1: Mean Collection Statistics Across All Indices

	num docs	num ents	max(nidf)	min(nidf)	words	phrases
Max	1795443	400094218	14.4	0.79	6924754	1816753
Min	1631840	377033932	14.31	0.7	5820360	1479404
Mean	1743419	385991502	14.37	0.74	6348646	1633945
Max-Min	163603	23060286	0.1	0.09	1104394	337349
Std Dev	43443	7832298	0	0	340602	112354

Table 2: Mean Term Statistics Across All Indices

	avg dist terms	min dist terms	max dist terms	avg terms	min terms	max terms
Max	239.53	0	38249	960.24	0	84337
Min	213.79	0	17788	804.15	0	61069
Mean	221.53	0	23642.73	857.42	0	68056.33
Max-Min	25.74	0	20461	156.09	0	23268
Std Dev	7	0	5774	46	0	6970

mum terms per document and maximum number of distinct terms per document. Most indexes have in the 60,000-70,000 range for maximum non-distinct terms, with two machines having a maximum of 83-84,000. The global maximum for number of non-distinct terms in a document was 84,337 terms. Considering that the average number of terms in a document was only 857 terms, this would suggest that long documents were spread across the cluster fairly evenly.

3.2 Querying

Querying the whole collection took one hour from the first machine to start until the last machine to end. Query performance was hampered by the fact that a configuration error caused the entire lexicon to be loaded into memory. This caused our memory footprint for the query engine to jump to around 1.6GB – almost the full extent of RAM. Since this meant that garbage collection had to be done for almost every posting list, each query took between 15 seconds and 5 minutes to perform, when retrieving the top 10,000 documents. Subsequent experiments with the lexicon in the proper format confirm that when our system is configured optimally, the memory footprint and query timings are much more reasonable, 160MB heap size and around 5 seconds per query. However this mistake confirms that it is, if not practical, possible to load the entire lexicon into memory at once for such a large collection.

Most query result sets on the individual machines provided the maximum number of scored documents each, 10,000. After the query engines on each machine finished, we copied the results to a single machine and used the UNIX sort utility to combine them, and then a short perl script to delete any results beyond the 10,000 highest ranked for each query. This process took 15 minutes for the entire result set. The simple vector-space and probability rankings with no relevance feedback returned enough results that only one query had few (less than 3,000) results.

4. CONCLUSIONS

In our experimental runs for this year's Terabyte track, we distributed the collection across a homogenous cluster and treated each node in the cluster has having an independent index. We then ran probabilistic and vector space models and merged the independent results on each node as if they were one result set in order to create two master result sets: one for the probabilistic model and one for the vector space model. In the future, we will experiment with creating a consolidated collection statistics structure and lexicon with the intent of providing the same relevance rankings for any document regardless of which sub-index it is found in. We will also experiment with different document distributions to see if term statistics change noticeably for a round-robin or random distribution of documents across our cluster and if this changes our effectiveness score. We will also work to make sure that relevance feedback is available in our next year's track.

5. ACKNOWLEDGEMENTS

Special thanks to Tristan Sloughter (IIT Information Retrieval Lab) for his work in distributing data and running tests tirelessly.

6. REFERENCES

[1] N. Craswell. *Methods for distributed information retrieval*. PhD thesis, The Australian National University, 2000.

[2] M. McCabe, A. Chowdhury, S. Beitzel, E. Jensen, M. SaeLee, D. Grossman, and O. Frieder. IIT TREC-9 - entity based feedback with fusion. In *Overview of the Ninth Text Retrieval Conference*, November 2000.

[3] S. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive. In *Proceeedings of the Seventh Text Retrieval Conference (TREC)* [5].

[4] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4), 1977.