

The TREC 2006 Terabyte Track

Stefan Büttcher
University of Waterloo
stefan@buetttcher.org

Charles L. A. Clarke
University of Waterloo
claclark@plg.uwaterloo.ca

Ian Soboroff
NIST
ian.soboroff@nist.gov

1 Introduction

The primary goal of the Terabyte Track is to develop an evaluation methodology for terabyte-scale document collections. In addition, we are interested in efficiency and scalability issues, which can be studied more easily in the context of a larger collection.

TREC 2006 is the third year for the track. The track was introduced as part of TREC 2004, with a single adhoc retrieval task. For TREC 2005, the track was expanded with two optional tasks: a named page finding task and an efficiency task. These three tasks were continued in 2006, with 20 groups submitting runs to the adhoc retrieval task, 11 groups submitting runs to the named page finding task, and 8 groups submitting runs to the efficiency task. This report provides an overview of each task, summarizes the results, and outlines directions for the future. Further background information on the development of the track can be found in the 2004 and 2005 track reports [4, 5].

For TREC 2006, we made the following major changes to the tasks:

1. We strongly encouraged the submission of adhoc manual runs, as well as runs using pseudo-relevance feedback and other query expansion techniques. Our goal was to increase the diversity of the judging pools in order to create a more re-usable test collection. Special recognition (and a prize) was offered to the group submitting the run contributing the most unique relevant documents to the judging pool.
2. The named page finding topics were created by task participants, with each group asked to create at least 12 topics.
3. The experimental procedure for the efficiency track was re-defined to permit more realistic intra- and inter-system comparisons, and to generate separate measurements of latency and throughput. In order to compare systems across various hardware configurations, comparative runs using publicly available search engines were encouraged.

2 The Document Collection

All tasks in the track use a collection of Web data crawled from Web sites in the `gov` domain during early 2004. We believe this collection (“GOV2”) contains a large proportion of the crawlable pages present in `gov` at that time, including HTML and text, along with the extracted contents of PDF, Word and postscript files. The collection is 426GB in size and contains 25 million documents. For TREC 2004, the collection was distributed by CSIRO, Australia, who assisted in its creation. For 2005 and 2006, the collection was distributed by the University of Glasgow.

3 Adhoc Retrieval Task

3.1 Task Description

An adhoc retrieval task investigates the performance of systems that search a static set of documents using previously-unseen topics. For each topic, participants create a query and generate a ranked list of documents. For the 2006 task, NIST created and assessed 50 new topics. An example is provided in Figure 1. In addition, the 99 topics created in 2004 and 2005 (topics 701-800) were re-used for automatic runs.

As is the case for most TREC adhoc tasks, a topic describes the underlying information need in several forms. The title field essentially contains a keyword query, similar to a query that might be entered into a Web search engine. The description field provides a longer statement of the topic requirements, in the form of a complete sentence or question. The narrative, which may be a full paragraph in length, supplements the other two fields and provides additional information required to specify the nature of a relevant document.

For the adhoc task, an experimental run consisted of the top 10,000 documents for each topic. To generate a run, participants could create queries automatically or manually from the topics. For a run to be considered *automatic* it must be created from the topics without any human intervention. All other runs are manual. Manual runs used only the 50 new topics; automatic runs used all 149 topics from 2004–2006.

For most experimental runs, participants could use any or all of the topic fields when creating queries from the topic statements. However, a group submitting any automatic run was required to submit at least one automatic run that used only the title field of the topic statement. Manual runs were encouraged, since these runs often add relevant documents to the evaluation pool that are not found by automatic systems using current technology. We offered a prize to the group with the run that returned the most unique relevant documents. Groups could submit up to five runs.

Runs were pooled by NIST for judging. The details of the pooling process differ substantially from previous years, and from previous TREC adhoc tasks, and are detailed in a separate section.

Assessors used a three-way scale of “not relevant”, “relevant”, and “highly relevant”. A document is considered relevant if any part of the document contains information which the assessor would include in a report on the topic. It is not sufficient for a document to contain a link that appears to point to a relevant Web page, the document itself must contain the relevant information. It was left to the individual assessors to determine their own criteria for distinguishing between relevant and highly relevant documents. For the purpose of computing effectiveness measures that require binary relevance judgments, the relevant and highly relevant documents are combined into a single “relevant” set.

```

<top>
<num> Number: 835

<title> Big Dig pork

<desc> Description:
Why is Boston's Central Artery project, also known as "The Big Dig",
characterized as "pork"?

<narr> Narrative:
Relevant documents discuss the Big Dig project, Boston's Central
Artery Highway project, as being a big rip-off to American taxpayers
or refer to the project as "pork". Not relevant are documents which
report fraudulent acts by individual contractors. Also not relevant
are reports of cost-overruns on their own.

</top>

```

Figure 1: *Adhoc Task Topic 835*

In addition to the top 10,000 documents for each run, we collected details about the hardware and software configuration used to generate them, including performance measurements such as total query processing time. For total query processing time, groups were asked to report the time required to return the top 20 documents, not the time to return the top 10,000. It was acceptable to execute a system twice for each run, once to generate the top 10,000 documents and once to measure the execution time for the top 20 documents, provided that the top 20 documents were the same in both cases.

3.2 Adhoc Pooling

Last year, we reported that the pools were increasingly dominated by documents containing the title words of topics, to the possible exclusion of other relevant documents [5, 6, 2]. This could lead to a test collection that is biased towards simple title-query-based retrieval approaches and that may not measure other systems fairly. We were able to observe this phenomenon directly in the AQUAINT collection due to the presence of a feedback run that retrieved many unique relevant documents. It seemed clear that it must also occur in GOV2, although we had no direct evidence in the form of missed relevant documents. For 2006, we took steps to gather data to answer the bias question for GOV2, including the active solicitation of manual runs in an effort to diversify the pools.

In addition, we required the adhoc runs to include the older topics to determine if the newer runs retrieved unusual amounts of unjudged documents for these topics. If so, it would provide more evidence of bias in the collection as well as data for analyzing the impact of that bias. However, since this activity was limited to automatic runs, we did not expect to see these runs fall far from the original qrels.

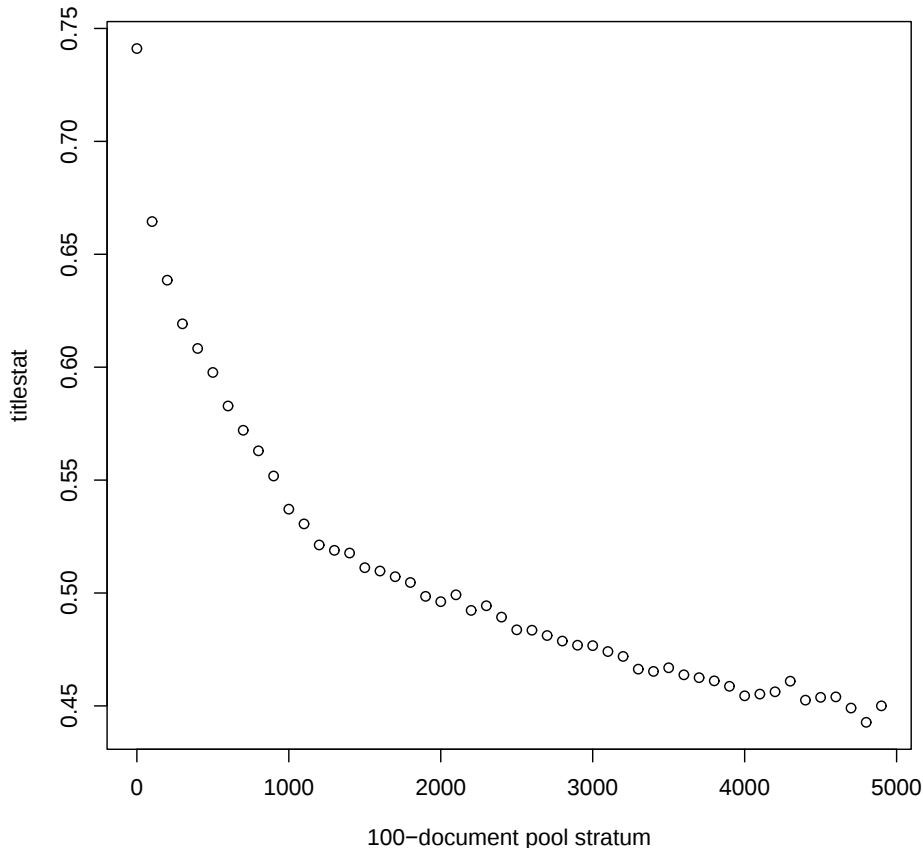


Figure 2: *Values for titlestat in 100-document strata of the pool.*

Lastly, we constructed three separate pools for the 2006 topics, two of which were used in the evaluation. The runs contributing to the pools are the same for all three: one automatic, one manual, and one efficiency task run per group. If a group only did manual or automatic runs, we took two of that type from that group.

The first pool is a traditional TREC pool to depth 50. This pool allows us to report traditional measures such as MAP and precision at fixed document cutoffs to some degree of accuracy, and thus provide some degree of continuity to track participants while experimental methods were tried.

The second pool is also a traditional TREC pool, but drawn starting at depth 400. This pool was motivated by the values of the *titlestat* measure, described by Buckley et al. [2]. The authors of that paper primarily computed the titlestat of judged relevant documents, but titlestat can be computed over any group of documents. For this application, we computed the titlestat of 100-document strata of the “complete” pool (that is, all documents pooled to depth 10,000). These titlestats are plotted to depth 5000 in figure 2. Whereas the pool from depths 1-100 has a titlestat of 0.74, at depths 400-500 the titlestat of the pool is just over 0.6. This pool starts at depth 400, but goes down to a different depth for each topic, such that the total size per topic for this plus the initial depth-50 pool is 1000 documents. While this pool was not used for the evaluation, the

relevance judgments allow us to see if relevant documents still occur frequently in lower-titlestat regions.

The third pool is a random sample, drawn in such a way as to attempt to gather more relevant documents deeply from topics where we expect them to occur. Using the relevance judgments from the depth-50 pool, we calculate the probability of relevance given the pool rank of a document. Using a simple linear fit based on experiments with last year’s pools, we then estimate the depth at which we will find a given ratio of relevant documents to pool size. We then draw a random sample of about 200 documents up to that target depth. This third pool varies in maximum pool depth from 57 to 1252 depending on how many relevant documents were found in the depth-50 pool for each topic.

The qrels from the third pool are not usable with MAP and other traditional measures. Instead, they are intended to be used with a new measure called *inferred average precision*, or *infAP* [7].¹ infAP estimates average precision based on a known pool where only a sample of documents in the pool are judged. The expected precision at rank k is

$$\frac{1}{k} \cdot 1 + \frac{k-1}{k} \left(\frac{P}{k-1} \cdot \frac{R+\epsilon}{R+N+\epsilon} \right)$$

where P is the number of documents in the pool, R is the number of known relevant documents above rank k , and N is the number of known nonrelevant documents above rank k . infAP is the average of these precisions at each relevant document. infAP is similar to bpref in that it is intended for incomplete judgments but differs in that it is a direct estimate of average precision based on a sample.

The official evaluation results report MAP (and other standard trec_eval measures) on the depth-50 pool, infAP on the random-sample pool, and for automatic runs MAP on all 149 terabyte track topics (where the AP scores for the 2006 topics are from the depth-50 pool.)

3.3 Adhoc Results

Table 1 provides an summary of the results obtained on the title-only automatic runs sorted by bpref. Only the best run from each group is shown. Figure 2 provides the same information for the manual runs. The first two columns of each table identify the group and run. The next three columns provide the values of three standard effective measures for each run: bpref [3], precision at 20 documents (p@20), and mean average precision (MAP). The sixth column provides values for the new infAP measure described above. The last two columns list the number of CPUs used to generate the run and the total query processing time.

The top-scoring automatics runs were generated using various retrieval methods, including Okapi BM25 and language modeling approaches. Many of these runs took features such as phrases, term proximity and matches in the title field into account during ranking. Of particular note is the prevalence of pseudo-relevance feedback, which substantially improved performance for most groups. On the other hand, none of the top-eight runs used anchor text, and only one used link analysis techniques.

A prize was offered to the group submitting the run containing the most unique relevant documents, excluding other runs from the same group. The prize (a NIST clock) was awarded to Chris Buckley of Sabir Research for the run sabtb06man1, which contributed 222 unique documents.

¹A technical note describing infAP can be found at http://trec.nist.gov/act_part/tracks/terabyte/inferredAP.pdf

Group	Run	bpref	p@20	MAP	infAP	CPUs	Time (sec)
uwaterloo.clarke	uwmtFadTPFB	0.4251	0.5570	0.3392	0.2999	1	964
umass.allan	indri06AlceB	0.4229	0.5410	0.3687	0.3157	1	38737
pekingu.yan	TWTB06AD01	0.4193	0.5150	0.3737	0.3224	4	56160
hummingbird.tomlinson	humT06xle	0.4172	0.5820	0.3452	0.2947	1	36000
ibm.carmel	JuruTWE	0.4002	0.5670	0.3506	0.2687	1	3375
uglasgow.ounis	uogTB06QET2	0.3995	0.5400	0.3456	0.2861	1	N/A
ecole-des-mines.beigbeder	AMRIMtp20006	0.3942	0.5170	0.3120	0.2994	2	68344
coveo.soucy	CoveoRun1	0.3886	0.5440	0.3296	0.2564	5	135
umilano.vigna	mg4jAutoV	0.3774	0.4510	0.2882	0.2765	4	3000
rmit.scholer	zetadir	0.3726	0.4800	0.3056	0.2599	2	466.5
umelbourne.ngoc-anh	MU06TBa2	0.3682	0.5130	0.3039	0.2549	1	25.25
uamsterdam.ilps	UAmsT06aTeLM	0.3528	0.4850	0.2958	0.2363	2	2394
dublincityu.gurrin	DCU05BASE	0.3509	0.5090	0.2695	0.2067	1	495
tsinghua.zhang	THUADALL	0.3432	0.4600	0.2858	0.2444	4	560
lowlands-team.deVries	CWI06DISK1ah	0.3361	0.4780	0.2770	0.2299	1	60.3
polytechnicu.suel	p6tbadt	0.3073	0.3920	0.2274	0.1972	1	60
max-planck.theobald	mpiirtitle	0.2849	0.4270	0.1805	0.1678	2	38
northeasternu.aslam	hedge0	0.2568	0.3460	0.1771	0.1388	4	110000
sabir.buckley	sabtb06at1	0.2434	0.3250	0.1361	0.1045	1	77
ualaska.fairbanks.newby	arscDomAlog	0.1463	0.0550	0.0541	0.0675	108	120000

Table 1: *Adhoc Results. Best automatic title-only run from each group, according to bpref.*

4 Named Page Finding Task

4.1 Task Description

Users sometimes search for a page by name. In such cases, an effective search system will return that page at or near rank one. In many cases there is only one correct answer. In other cases, any document from a small set of “near duplicates” is correct. The objective of the task, therefore, is to find a particular page in the GOV2 collection, given a topic that describes it. For example, the query “**Apollo 11 Mission**” would be satisfied by NASA’s history page on the first moon landing.

Named page topics were created by track participants through a purpose-built Web interface to the Wumpus search engine² and hosted at the University of Waterloo. Participants were asked to imagine they were using a search engine to locate an interesting page that they found once but couldn’t quite remember where it was. Their goal was to identify interesting, “bookmark-worthy” pages, that they thought they might want to go back and find again. Once such a page was found, they were to give it a name such as they might assign to a bookmark. The name was to approximate what they might type into a Web search engine to locate that page again.

Participants could identify interesting pages in one of two ways. One was to request random pages from the search engine, and to keep looking at random pages until one struck their fancy.

²<http://www.wumpus-search.org/>

Group	Run	bpref	p@20	MAP	infAP	CPUs	Time (sec)
uwaterloo.clarke	uwmtFmanual	0.4785	0.7030	0.4246	0.3503	1	20000
sabir.buckley	sabtb06man1	0.4104	0.6070	0.2666	0.2161	1	21600
pekingu.yan	TWTB06AD02	0.4089	0.5070	0.3152	0.2749	4	9625
rmit.scholer	zetaman	0.3976	0.5290	0.2873	0.2369	2	307
umilano.vigna	mg4jAdhocBV	0.3944	0.4930	0.2822	0.2465	4	610
umelbourne.ngoc-anh	MU06TBa1	0.3900	0.5420	0.2927	0.2431	8	15.50
ecole-des-mines.beigbeder	AMRIMtpm5006	0.3793	0.4390	0.2705	0.2702	2	39032
ibm.carmel	JuruMan	0.3570	0.5190	0.2754	0.2410	1	60
northeasternu.aslam	hedge30	0.3180	0.5110	0.2561	0.1942	4	18000
max-planck.theobald	mpiirmanual	0.3041	0.4810	0.1981	0.1692	2	25
ualaska.fairbanks.newby	arscDomManL	0.1202	0.0400	0.0351	0.0511	108	150000

Table 2: *Adhoc Results (manual runs), sorted by bpref.*

Another was to search for subjects of interest to the participant, and to look through the search results until something worth keeping was found.

Participants were instructed to make each page’s name specific to that page. To check this, they were requested to perform a search with the name as a query, and to check to see if other pages came up which could take the same name. The named page itself did not need to appear in these search results, although it was acceptable if it did. The purpose of this check search was to weed out similar (but not near-duplicate) pages that might need to be distinguished in order to obtain a good named page topic. Near-duplicates of the page, which differ only in formatting or by trivial content changes, were permitted.

When evaluating the submitted runs, we identified these near-duplicates using NIST’s implementation of Bernstein and Zobel’s DECO algorithm [1]. We ran DECO on the top 100 retrieved documents from all submitted named page runs, identified near-duplicates of the known targets, and manually checked those for relevance. Near-duplicates are treated as equivalent to the original page and are included in the qrels file.

4.2 Named Page Finding Results

Figure 3 summarizes the results of the named page finding task. The performance of the runs is evaluated using three metrics:

- **MRR:** The mean reciprocal rank of the first correct answer.
- **% Top 10:** The proportion of queries for which a correct answer was found in the first 10 search results.
- **% Not Found:** The proportion of queries for which no correct answer was found in the results list.

The figure lists the best run from the each group by MRR. In addition, the figure indicates the runs that exploit link analysis techniques (such as pagerank), anchor text, and document structure (such as giving greater weight to terms appearing in titles).

Group	Run	MRR	% Top 10	% Not Found	CPUs	Time (sec)	Links?	Anchors?	Structure?
umass.allan	indri06NsdP	0.512	69.6	13.8	6	10860	Y	Y	Y
uglasgow.ounis	uogTB06MP	0.466	65.2	12.7	1		N	Y	Y
coveo.soucy	CoveoNPRun2	0.431	59.1	19.9	5	235	N	N	Y
tsinghua.zhang	THUNPNOSTOP	0.430	64.1	16.0	2	3240	N	Y	N
hummingbird.tomlinson	humTN06dpl	0.408	56.9	13.3	1	4600	N	N	Y
umelbourne.ngoc-anh	MU06TBn5	0.397	62.4	13.8	1	50	N	Y	N
rmit.scholer	zetnpft	0.389	54.7	19.3	2	2001	N	N	Y
uwaterloo.clarke	uwmtFnpsRR1	0.386	54.7	18.8	1	1149	Y	Y	Y
uamsterdam.ilps	UAmsT06n3SUM	0.363	55.2	23.8	2	2545	N	Y	Y
cas-ict.wang	icttb0603	0.337	44.2	28.7	1	427	N	N	Y
pekingu.yan	TWTB06NP02	0.238	34.3	44.2	4	3240	N	N	Y

Table 3: *Named Page Finding Results. Best run from each group, according to MRR.*

5 Efficiency Task

5.1 Task Description

The efficiency task extends both the adhoc task and the named page finding task, providing a vehicle for discussing and comparing efficiency and scalability issues in IR systems by defining better methodology to determine query processing times.

Two weeks before the new topics for the adhoc task were made available, NIST released a set of 100,000 efficiency topics. These topics were extracted from the logs of a commercial Web search engine. Because an analysis of last year’s 50,000 efficiency topics, which also had been extracted from a Web search engine log, had shown that the topics did not match GOV2 very well and consequently could be processed much faster than the adhoc topics, this year we made sure that each query in the efficiency topic set:

- had produced at least one clickthrough to .gov in the Web search engine, and
- matched at least 20 documents in GOV2 (Boolean OR).

After creating a set of representative topics in this way, the title fields of the adhoc topics (751–850) and the named page finding topics (NP601–NP872, NP901–NP1081) from this year’s and last year’s Terabyte track were seeded into the topic set, but were not distinguished in any way. Figure 3 provides some examples from the resulting topic set. Participating groups were required to process these topics automatically; manual runs were not permitted.

The efficiency topic set was distributed in 4 separate files, representing 4 independent query streams. Groups were required to process queries within the same stream sequentially and in the order in which they appear in the topic file. Processing of each query in a stream was to be

68964:easy to do science projects for 5th grade
 68965:who to contact when civil rights are violated
 68966:ergonomic courses illinois
 68967:big dig pork
 68968:signs of a partial seizure
 68969:food at jfk
 68970:natural gas power point
 68971:va home care portland oregon
 68972:lexapro package insert

Figure 3: *Efficiency Task Topics 68964 to 68972*

completed before processing of the next query was started. Queries from different query streams could be processed concurrently or interleaved in any arbitrary order. The existence of independent query streams allowed systems to take better advantage of parallelism and I/O scheduling.

Each participating group ran their system on the entire topic set (all four streams), reporting the top 20 documents for each topic, the average per-topic response time (referred to as query processing *latency*), and the total time between reading the first topic and writing the last result set (used to calculate query throughput). The total time was recorded without taking into account system startup times. By processing all queries strictly sequentially, latency was minimized. A group was able to choose, however, to process queries from different streams in parallel in order to make better use of parallelism and to increase their system’s query throughput.

In general, document retrieval systems can employ two different kinds of parallelism: intra-query and inter-query. With intra-query parallelism, the same query is processed on multiple CPUs in parallel, for example by splitting the document collection into equal parts and distributing these parts among different machines. Intra-query parallelism improves both latency and throughput, although not necessarily in a linear fashion. Inter-query parallelism, on the other hand, refers to the situation in which multiple queries are being processed at the same time. It can be used to increase query throughput, usually in a linear or near-linear way, but does not improve query latency. Distributing the 100,000 efficiency topics in four independent streams was meant to explicitly encourage inter-query parallelism and to allow groups to study latency/throughput trade-offs.

One of the goals of the Terabyte track is to be able to compare different approaches to high-performance information retrieval and to evaluate them quantitatively. The validity of direct comparisons between groups, however, is limited by the range of hardware used, which varies from desktop PCs to supercomputers. Last year, we tried to overcome this issue by applying some informal normalizations, based on the number of CPUs and the total cost of a system. Those attempts were only partially successful. In order to obtain more reliable inter-group performance comparisons, participants this year were encouraged to submit at least one run that was conducted in a single-CPU configuration, with all queries being processed sequentially. They were also encouraged to download special TREC versions of the three open-source information retrieval systems

- Indri (<http://www.lemurproject.org/indri/>),
- Wumpus (<http://www.wumpus-search.org/>), and
- Zettair (<http://www.seg.rmit.edu.au/zettair/>);

Run				Latency			Throughput			Effectiveness	
	Number of CPUs	System cost (USD)	Query streams	measured (ms/query)	CPU-normalized	cost-normalized	measured (queries/s)	CPU-normalized	cost-normalized	P@20 (801-850)	MRR (901-1081)
CWI06DIST8	16	6,400	4	13	211	85	185.5	11.6	29.0	0.4680	0.181
CWI06MEM4	4	10,000	4	80	322	805	48.7	12.2	4.9	0.4720	0.190
humTE06i3	1	5,000	1	1,680	1,680	8,400	0.6	0.6	0.1	0.3690	0.123
humTE06v2	1	5,000	1	4,630	4,630	23,150	0.2	0.2	0.0	0.4290	0.373
mpiiotopk2p	2	5,000	4	29	57	143	35.0	17.5	7.0	0.4330	0.280
mpiiotopkpar	2	5,000	4	74	148	369	13.6	6.8	2.7	0.4280	0.291
MU06TBy6	1	500	4	55	55	28	18.2	18.2	36.4	0.4890	0.271
MU06TBy2	1	500	1	229	229	114	4.4	4.4	8.8	0.5050	0.256
p6tbep8	1	1,400	1	109	109	153	9.1	9.1	6.5	0.3890	0.254
p6tbeb	1	1,400	1	167	167	234	6.0	6.0	4.3	0.4540	0.244
rmit06effic	2	4,000	1	2,202	4,404	8,808	0.5	0.2	0.1	0.4650	0.258
THUTeraEff02	4	2,000	1	534	2,136	1,068	1.9	0.5	0.9	0.1500	0.222
THUTeraEff01	4	2,000	1	808	3.232	1,616	1.2	0.3	0.6	0.3920	0.246
uwmtFdcp03	1	1,800	1	13	13	23	80.0	80.0	44.4	0.4110	0.164
uwmtFdcp12	1	1,800	1	32	32	58	31.4	31.4	17.4	0.4790	0.219

Table 4: *Efficiency Results. Best run (according to P@20) and fastest run (according to average query latency) from each participating group. Effectiveness is reported for the the adhoc (P@20) and the named page finding (MRR) topics from 2006, not for the 2005 topics also present in the efficiency topics.*

to compile them, build an index for GOV2, and run 10,000 queries (a subset of the 100,000 efficiency topics) through these systems. For each such comparative efficiency run, participants were required to report the total time to process all queries and the total CPU consumed, i.e., the time during which the CPU was busy processing queries and not waiting for the hard drive, for instance.

Our incentive was that this data could be used to find out whether an inter-group performance comparison, across different hardware configurations, is feasible at all and also how efficiency numbers need to be normalized in order to obtain a fair comparison of two systems running on different hardware.

We would like to point out that the versions of the three systems we used for comparative reasons were modified and re-packaged for this specific purpose, without special support from their developers. It is therefore unlikely that the efficiency numbers that were obtained reflect the true performance of the systems. They were used exclusively to determine the performance of the hardware they were run on.

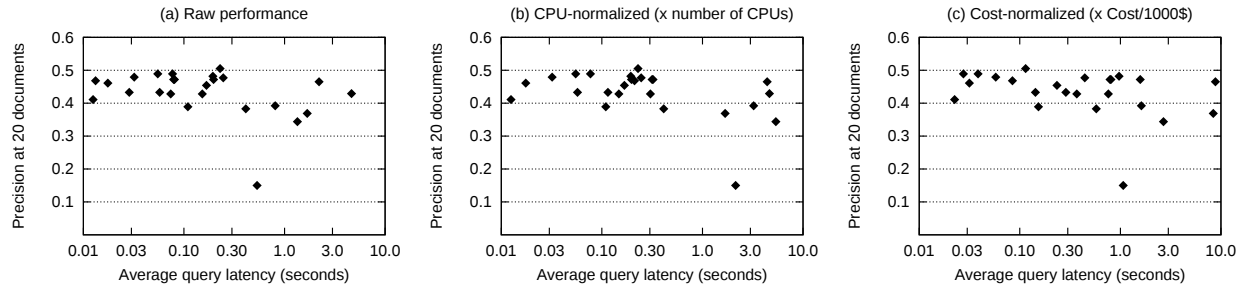


Figure 4: *Efficiency (average latency for the 100,000 efficiency topics) and effectiveness ($P@20$ for adhoc topics 801–850) of all 25 runs submitted for the efficiency task.*

5.2 Efficiency Results and Discussion

The efficiency results for the 100,000 efficiency topics are shown in Table 4. For each group, the best run (according to mean precision at 20 documents, for adhoc topics 801–850) and the fastest run (according to average query latency) are summarized. Both performance and precision vary greatly among the systems. The two fastest runs exhibit an average latency of 13 ms per query, while the slowest run consumed almost 5 seconds. $P@20$, on the other hand, varies quite substantially as well, between 0.150 and 0.505.

Although groups were allowed to have their system process queries from the four separate query streams in parallel, most groups chose not to do so. Only 5 out of the 25 efficiency runs processed queries in parallel. For some of these runs, however, the gains achieved from processing queries in parallel are tremendous, leading to a query throughput of up to 185 queries per second in the case of CWI06DIST8.

The efficiency measures reported in Table 4 include throughput and latency, both in their raw form (as measured by the respective group) as well as normalized, applying the same ad-hoc performance normalizations (CPU-normalized and cost-normalized) as last year:

- the CPU-normalized query latency is the real query latency, multiplied by the total number of CPUs in the system;
- the cost-normalized query latency compares each run with a run conducted on a hypothetical computer system costing 1,000 USD; thus, the latency of a run conducted on a \$5,000 computer was multiplied by 5.

Efficiency vs. effectiveness trade-off plots, both normalized and non-normalized, for all 25 efficiency runs are shown in Figure 4.

The validity of calculating CPU-normalized query latency is actually somewhat questionable. Using multiple CPUs in parallel only decreases latency in the case of intra-query parallelism, not in the case of inter-query parallelism. Nonetheless, because we do not know exactly which type of parallelism a group employed in their runs, we decided to apply CPU normalization to both efficiency measures, throughput and latency.

Unfortunately, however, neither CPU normalization nor cost normalization are completely satisfying. The number of CPUs in a system does not say anything about the performance of these CPUs. The total cost of a computer, on the other hand, might include components like graphics cards and additional hard drives that were not used in a run at all.

Group	Comp. Run	Indri	Wumpus	Zettair
umass.allan	n/a	7.38/6.04/32-bit	0.48/0.32/32-bit	1.83/0.92/32-bit
max-planck.theobald	mpiitopk2		0.23/0.18/64-bit	
rmit.scholer	rmit06effic	5.89/5.24/32-bit	0.39/0.30/32-bit	1.50/1.10/32-bit
lowlands-team.deVries	CWI06DISK1	4.64/4.17/64-bit	0.22/0.14/64-bit	0.93/0.81/32-bit
uwaterloo-clarke	uwmtFdcpl2	4.49/3.02/64-bit	0.24/0.11/64-bit	1.62/0.62/32-bit

Table 5: *Comparative efficiency runs using Indri, Wumpus, and Zettair. Each field contains the average query latency in seconds (left), the average CPU time per query in seconds (middle), and the CPU type (right), either 32-bit or 64-bit.*

In order to obtain more reliable comparative performance results, we analyzed the efficiency numbers reported as part of the comparative efficiency runs conducted with Indri, Wumpus, and Zettair. Comparative runs were submitted by 4 out of 8 participating groups. In addition, Trevor Strohman from the University of Massachusetts (umass.allan) submitted three comparative runs in order to help us with our data sparseness problem. The results are shown in Table 5.

When examining the reported efficiency numbers, we noticed that, while performance figures were largely consistent between two different 32-bit systems (for both umass.allan and rmit.scholer, for instance, Wumpus is about 3.8 times faster than Zettair, while Indri is about 4 times slower than Zettair), the situation is different when comparing 32-bit hardware with 64-bit hardware. On uwaterloo-clarke’s hardware, Wumpus is about 6.8 times faster than Zettair, while Indri is only 2.8 times slower than Zettair.

The large discrepancy between Wumpus and Zettair is due to Wumpus being designed for 64-bit and using 64-bit integer arithmetic throughout. When compiled for 32-bit, all 64-bit integer instructions have to be simulated by sequences of 32-bit integer instructions, a transformation that is very costly. Zettair, on the other hand, uses 32-bit integer arithmetic and is compiled for a 32-bit architecture. Therefore, it does not experience the same slowdown when executed on a 32-bit computer. When uwaterloo-clarke recompiled Wumpus for 32-bit and ran it on their 64-bit hardware, this discrepancy almost vanished, and Wumpus was only 5 times faster than Zettair.

However, the non-proportional performance difference between the three retrieval systems when moving between different hardware configurations is not only because of CPU issues, but also because of different hard drive performance in the individual computer systems: uwaterloo-clarke stored index structures on a single disk, while umass.allan stored the index on a 3-disk RAID, lowlands-team.deVries on a 10-disk RAID, and rmit.scholer even on a 12-disk RAID. Because Indri, Wumpus, and Zettair produce inverted files of different size, the hard drive configuration has different effects on the three systems. For example, while Wumpus exhibits about the same performance on the hardware configurations used by lowlands-team.deVries and uwaterloo-clarke (220 vs. 240 ms per query), Zettair is 75% faster on lowlands-team.deVries’s hardware than on uwaterloo-clarke’s (930 vs. 1620 ms).

Unfortunately, the effect of different hard disk configurations cannot be eliminated by comparing CPU times, either. A system that uses software RAID, for instance, usually exhibits higher CPU utilization than a system using hardware RAID.

Despite these irregularities, we tried to come up with a tentative performance normalization procedure that allows us to compare the performance of different retrieval systems across hardware configuration boundaries. Because all four groups participating in the comparative efficiency task

Run	HW performance	Latency	Normalized latency	P@20	MRR
CWI06DISK1	CPU: 2.23, Total: 2.23	197 ms	439 ms ... 440 ms	0.4720	0.196
mpiiotopk2	CPU: 1.71, Total: 2.15	75 ms	128 ms ... 161 ms	0.4330	0.280
rmit06effic	CPU: 1.05, Total: 1.26	2,202 ms	2,304 ms ... 2,768 ms	0.4650	0.258
uwmtFdcp12	CPU: 2.89, Total: 2.01	32 ms	64 ms ... 92 ms	0.4790	0.219

Table 6: *Normalized efficiency based on the true performance of the underlying hardware configuration (estimated through Wumpus). Hardware performance is given relative to umass.allan (2.6 GHz Pentium IV with 2 GB RAM and a 3-way software RAID-0).*

submitted a comparative run using Wumpus, we chose to use Wumpus to establish the true performance of the underlying hardware and to normalized the latency numbers reported by the groups based on the performance estimate obtained through Wumpus. For each group, two performance estimates were obtained, one based on CPU time, the other based on average query latency. This led to a normalized efficiency interval instead of a single efficiency number. The results are shown in Table 6. Because of all the difficulties explained above, the outcome of the normalization process should be taken with a grain of salt.

If we have learned anything from our attempt to conduct a fair performance comparisons of different retrieval systems, then it is the insight that such a comparison is incredibly difficult, if not impossible, to achieve. The assumption that all document retrieval systems are relatively similar to each other and thus have similar performance characteristics (and by that we do not mean raw latency or throughput values) does not hold. It is entirely possible that moving to a different hardware configuration improves the performance of retrieval system A while depleting the performance of system B, as documented by Table 5

6 The Future of the Terabyte Track

2006 is the final year of the Terabyte Track in its current form. After three years, we have a reasonable number of topics and judgments, and we cannot see significant value in another year of similar experiments on the same collection.

In the future, we hope to resurrect the track with a substantially larger collection and an renewed focused on Web retrieval. Along with the standard adhoc and named page finding tasks, we plan to examine problems such as Web spam filtering and snippet extraction.

Acknowledgments

We thank everyone who helped to establish and operate the track over the past three years, particularly Yaniv Bernstein, Nick Craswell, David Hawking, Doug Oard, Falk Scholer and Ellen Voorhees.

References

- [1] Yaniv Bernstein and Justin Zobel. A scalable system for identifying co-derivative documents. In *Proceedings of the Symposium on String Processing and Information Retrieval*, pages 55–67, Padova, Italy, 2004.
- [2] Chris Buckley, Darrin Dimmick, Ian Soboroff, and Ellen Voorhees. Bias and the limits of pooling. In *Proceedings of SIGIR 2006*, 2006.
- [3] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32, Sheffield, UK, 2004.
- [4] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 Terabyte Track. In *Proceedings of the Thirteenth Text REtrieval Conference*, Gaithersburg, MD, November 2004. NIST Special Publication 500-261. See `trec.nist.gov`.
- [5] C. L. A. Clarke, F. Scholer, and I. Soboroff. The TREC 2005 Terabyte Track. In *Proceedings of the Fourteenth Text REtrieval Conference*, Gaithersburg, MD, November 2005. NIST Special Publication 500-266. See `trec.nist.gov`.
- [6] Ellen M. Voorhees. Overview of the TREC 2005 robust retrieval track. In *Proceedings of the Fourteenth Text REtrieval Conference*, Gaithersburg, MD, November 2005. NIST Special Publication 500-266. See `trec.nist.gov`.
- [7] Emine Yilmaz and Javed Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of CIKM 2006, to appear*, 2006.