Effective Smoothing for a Terabyte of Text

Jaap Kamps[1,2]

[1] Informatics Institute, University of Amsterdam
[2] Archives and Information Studies, Faculty of Humanities, University of Amsterdam
http://ilps.science.uva.nl/

Abstract: As part of the TREC 2005 Terabyte track, we conducted a range of experiments investigating the effects of larger collections. Our main findings can be summarized as follows. First, we tested whether our retrieval system scales up to terabyte-scale collections. We found that our retrieval system can handle 25 million documents, although in terms of indexing time we are approaching the limits of a non-distributed retrieval system. Second, we hoped to find out whether results from earlier Web Tracks carry over to this task. For known-item search we found that, on the one hand, indegree and URL priors did not promote retrieval effectiveness, but that, on the other hand, the combination of different document representations improved retrieval effectiveness. Third, we investigated the role of smoothing for collections of this size. We found that larger collections require far less smoothing, especially for the adhoc task using very little smoothing leads to substantial gains in retrieval effectiveness.

1 Introduction

As part of the TREC 2005 Terabyte track, we conducted a range of experiments investigating the effects of larger collections. First, we want to test whether our retrieval system scales up to 25 million documents. Second, we hope to find our whether results from earlier Web Tracks carry over to this task. Third, we want to investigate the role of smoothing for collections of this size.

We submitted runs for two of the Terabyte track's tasks: the adhoc task, and the named page finding task. In addition to the submitted runs, we also discuss post-submission results for the efficiency task. Furthermore, we discuss a range of post-submission experiments that further clarify the role of smoothing, especially for adhoc retrieval on terabyte-scale collections..

The rest of this paper is organized as follows. In Section 2, we detail the experimental set-up for the three tasks in the Terabyte track. In Section 3, we discuss our results, broken down over the efficiency task (§3.1); the adhoc task (§3.2); and the named page finding task (§3.3). In Section 4, we zoom in on a set of experiments on the amount of smoothing for terabyte-sized collections. Finally, we summarize our findings in Section 5.

2 Experiments

Our retrieval system is based on the Lucene engine with a number of home-grown extensions [2, 6].

2.1 Indexes

The Terabyte track uses the GOV2 test collection, containing 25,205,178 documents (426 Gb uncompressed). We created three separate indexes for (1) the full documents, (2) the text in the title tags, (3) the anchor-texts pointing toward the document. For the anchor-texts index, we ignored relative links and only extracted full links. We normalized URLs, and did not index repeated occurrences of the same anchor-text. This is similar to our earlier experiments in the TREC Web track [4, 5]. As to tokenization, we removed HTML-tags, punctuation marks, applied case-folding, and mapped marked characters into the unmarked tokens. We used the Snowball stemming algorithm [7].

We created a single, non-distributed index for the collection. The size of our full-text index is 61 Gb. Building the full-text index (including all further processing) took a massive 15 days, 6 hours, and 21 minutes.

2.2 Retrieval models

For our ranking, we use either a vector-space retrieval model or a language model. Our vector space model is the default similarity measure in Lucene [6], i.e., for a collection D, document d and query q:

$$sim(q,d) = \sum_{t \in q} \frac{tf_{t,q} \cdot idf_t}{norm_q} \cdot \frac{tf_{t,d} \cdot idf_t}{norm_d} \cdot coord_{q,d} \cdot weight_t,$$

where

$$tf_{t,X} = \sqrt{\text{freq}(t,X)}$$

$$
\begin{aligned}
idf_t &= 1 + \log \frac{|D|}{\text{freq}(t,D)} \\
norm_q &= \sqrt{\sum_{t \in q} tf_{t,q} \cdot idf_t^2} \\
norm_d &= \sqrt{|d|} \\
coord_{q,d} &= \frac{|q \cap d|}{|q|}
\end{aligned}
$$

Our language model is an extension to Lucene [2], i.e., for a collection D, document d and query q:

$$P(d|q) = P(d) \cdot \prod_{t \in q} ((1-\lambda) \cdot P(t|D) + \lambda \cdot P(t|d)),$$

where

$$
\begin{aligned}
P(t|d) &= \frac{tf_{t,d}}{|d|} \\
P(t|D) &= \frac{\text{doc\_freq}(t,D)}{\sum_{t' \in D} \text{doc\_freq}(t',D)} \\
P(d) &= \frac{|d|}{\sum_{d' \in D} |d'|}
\end{aligned}
$$

The standard value for the smoothing parameter λ is 0.15.

2.3 Official runs

We submitted six runs in total, using only the short topic statement in the title. For the adhoc task, we submitted the following two runs:

UAmsT05aTeVS Vector space model on the full-text index.

UAmsT05aTeLM Language model (λ = 0.15) on the full-text index.

For the named page finding task, we submitted four runs. We submitted a plain language model run:

UAmsT05nTeLM Language model (λ = 0.70) on the full-text index.

We also experimented with web-centric priors [3]. First, we assumed that pages with more inlinks are more likely to be relevant. Since our implementation of the language model calculates the logs of the probabilities, we took the exponent of the retrieval score, and multiplied it with the root of the indegree. We used the incomplete indegree scores we obtained from the anchor-text index. Second, we assumed that pages with shorter URLs are more likely to be relevant. We calculated the number of components in the domain and file path of the URL, e.g, `trec.nist.gov/act_part/act_part.html` has 3 (domain) plus 2 (file path) components. Again, we took the exponent of the retrieval score, and multiplied it with the reciprocal of the length of the URL.

UAmsT05nTind Language model (λ = 0.70) on the full-text index, with an indegree prior.

UAmsT05nTurl Language model (λ = 0.70) on the full-text index, with a URL prior.

Finally, we have not yet implemented a proper mixture language model incorporating different document representations. Instead, we combined separate runs made on the different full-text, anchor-text, and titles indexes.

UAmsT05n3SUM CombSUM of language model (λ = 0.70) runs on the full-text index (relative weight 0.8), anchor-text index (relative weight 0.8), and titles index (relative weight 0.8).

3 Results

3.1 Efficiency task

We created a run for the efficiency task as a post-submission experiment. Table 1 shows the total and average query processing times for the 50,000 efficiency task topics. We used

Task	#Topics	Model	Total		Avg.Q
Efficiency	50,000	VS	23,976	(6h39m36s)	0.480
Adhoc	50	VS	43	(43s)	0.862
Adhoc	50	LM	798	(13m18s)	15.962
Named Page	272	VS	594	(9m54s)	2.184
Named Page	272	LM	12,180	(3h23m)	44.781

Table 1: Performance measurements in seconds for max. 20 results per topic using the full-text index.

a non-dedicated, dual processor machine running Linux with the retrieval system running as a single Java process with a heap size of 1 Gb. For the efficiency task we used the vector-space model. Total processing time for the 50,000 queries was 6 hours and 39 minutes. On average, it took 0.480 seconds to produce the top 20 results for a single query. For comparison, we also list the system performance for the other Terabyte tasks. In terms of effectiveness, the efficiency task run is identical to ad hoc task run `UAmsT05aTeVS` discussed below. Its precision at rank 20 is 0.3710.

3.2 Adhoc task

There are in total 50 adhoc task topics. The number of relevant documents per topics varies from 4 to 559, with an average of 208 and a median 171. Table 2 shows the results for the adhoc task. We see an interesting compari-

UAmsT05	#rel_ret	map	bpref	recip_rank	P@10	P@20
...aTeVS	**5717**	**0.1996**	**0.2596**	0.4437	**0.3800**	**0.3710**
...aTeLM	4180	0.1685	0.2083	**0.6023**	**0.3800**	0.3460

Table 2: Results for the adhoc task.

son between the two retrieval models. First, we see that the vector-space model is superior on the overall measures (map and bpref). Second, we see that the language model is superior at early precision (recip_rank), but that this waters down quickly (precision is equal at rank 10, and less at rank 20). The outcome deviates from results on the training data—the Terabyte track 2004 adhoc task topics—, where the language model outperformed the vector space model with a map score of 0.1562 versus 0.1413. Below, we will further zoom in on the language model and experiment with the amount of smoothing.

3.3 Named page finding task

In total there are 252 named page finding topics (20 topics have been deemed adhoc topics, and have been retracted from the qrels). The minimal number of relevant documents per topic is 1 and the maximum is 4,525. For 187 topics there is a unique relevant page, the few topics with thousands of relevant pages are caused by page-duplicates in the collection. This leads to a skewed distribution with a mean of 47 and a median of 1 relevant page. Table 3 shows the results for the named page finding task. We make a number

UAmsT05	recip_rank	top 10		not found	
...TeLM	0.3364	112	44.44%	58	23.02%
...Tind	0.2649	99	39.29%	58	23.02%
...Turl	0.3251	115	45.63%	58	23.02%
...3SUM	**0.3653**	**123**	48.81%	**57**	22.62%

Table 3: Results for the named page finding task.

of observations. First, the indegree prior results in a loss of performance. Second, the URL prior leads to mixed results: a loss of mean reciprocal rank, but a gain in the number of topics with the relevant page in the top 10. Third, the combination run leads to improved performance on all measures.

The success of the combination run shows the value of different document representations. On the Web Track data, mixture language models proved far more effective than straightforward run combination [4, 5]. This may also explain, in part, the mixed results for the link and URL priors. Other factors such as the incompleteness of the extracted links may also play an important role.

4 Smoothing experiments

In the language modeling framework, smoothing plays an important role: it helps to overcome data-sparseness, it introduces an inverted document frequency effect, and it expresses the relative importance of query terms [8]. In practice, smoothing is also a handle to tune a run toward recall (much smoothing) or precision (little smoothing). It is known that collection size is a factor influencing precision measures [1]. Hence, collection size may also be a factor influencing the amount of smoothing needed in the language modeling framework. Here, we focus on linear or Jelinek-Mercer smoothing, and investigate the effect of varying the smoothing parameter.

4.1 Named page finding task

First, we focus on the named page finding task. Since finding a 'unique' page requires precision rather than recall, we choose a relatively high value for the smoothing parameter (i.e., $\lambda = 0.7$). Table 4 shows the results while varying the smoothing parameter over the interval between 0 and 1. We make a few observations. As expected, we see that

λ	recip_rank	top 10		not found	
0.0	0.0000	0	0.00%	252	100.00%
0.1	0.1684	57	22.62%	135	53.57%
0.2	0.2124	78	30.95%	107	42.46%
0.3	0.2417	83	32.94%	90	35.71%
0.4	0.2753	98	38.89%	79	31.35%
0.5	0.3046	103	40.87%	70	27.78%
0.6	0.3158	110	43.65%	64	25.40%
0.7	0.3364	112	44.44%	58	23.02%
0.8	0.3447	115	45.63%	57	22.62%
0.9	**0.3557**	**118**	46.83%	**54**	21.43%
1.0	0.3436	117	46.43%	61	24.21%

Table 4: Smoothing for the named page finding task using the full-text index.

the named page finding topics do not require much smoothing. In fact, as long as we put some weight on the collection model, the less smoothing the better. This is in contrast with results on the Web Track data, where performance actually drops at the highest values of the smoothing parameter.

4.2 Adhoc task

Next, we focus on the adhoc task. Since adhoc topics require a delicate balance between precision and recall, we choose the standard relatively low value for the smoothing parameter (i.e., $\lambda = 0.15$). Table 5 shows the results while varying the smoothing parameter over the interval between 0 and 1. A few observations present themselves. We see that performance increases if we apply less smoothing. In fact, the gain is substantial; already the improvement for $\lambda = 0.2$ is statistically significant (99.9%, one tailed) over $\lambda = 0.15$. In sum, the adhoc task evaluated by mean average precision behaves like an early precision task.

4.3 On Collection Size and Smoothing

We conduct an initial, exploratory experiment to study the effect of collection size on smoothing. First, we took the list of documents-ids randomly removed 50% of the documents. By doing this repeatedly, we obtained six samples containing 100%, 50%, 25%, 12.5%, 6.25% and 3.125% of the original collection, where documents are always also contained in

λ	MAP	B-Pref	Prec@10	Prec@20
0.0	0.0002	0.0045	0.0020	0.0010
0.1	0.1467	0.1847	0.3620	0.3210
0.2	0.1856	0.2272	0.4120	0.3650
0.3	0.2138	0.2576	0.4600	0.4050
0.4	0.2355	0.2783	0.4900	0.4490
0.5	0.2540	0.2943	0.5060	0.4830
0.6	0.2707	0.3101	0.5380	0.5110
0.7	0.2863	0.3243	0.5420	0.5320
0.8	0.2998	0.3366	0.5620	0.5420
0.9	**0.3107**	**0.3460**	0.5680	0.5530
1.0	0.2972	0.3437	**0.5840**	**0.5600**

Table 5: Smoothing for the adhoc task using the full-text index.

larger samples. Table 6 shows the resulting sample sizes, in number of documents, as well as the number of relevant documents (for any of the Adhoc topics) remaining. For all

Percentage	Size in Docs	Relevant Docs
100.000%	25,205,179	45,291
50.000%	12,602,589	22,539
25.000%	6,301,294	11,356
12.500%	3,150,647	5,641
6.250%	1,575,323	2,778
3.125%	787,661	1,375

Table 6: Samples of the GOV2 collection.

50 Adhoc topics, there is at least one relevant document in all samples. Second, we retrieved the top 10,000 documents for each of the topics, and for eleven values of λ in the range $[0,1]$. Then, for each of the samples, we restricted both the run and the qrels by removing document no longer present in the sample. This allows us to evaluate the performance on each of the samples of the total collection.

Figure 1(top) shows precision at 10 scores of the six samples for all values of the smoothing parameter. We see a solid decline in the scores when the samples get smaller. This in line with the expectation that early precision scores increase with collection size [1]. The scores over different values of the smoothing parameter gets less steep when the sample sizes decrease, but the optimal value of λ remains high. What about mean average precision? Figure 1(middle) has the MAP scores of the six samples, again over all values of the smoothing parameter. Here, we see almost the same performance over the smoothing parameter for the four samples larger than 10% of the whole collection. For smaller samples, the scores are somewhat higher, but this may be due to a favorable choice of our sampling strategy. As for the smoothing, the optimal values stay markedly in the highest region. As similar picture arises for the binary preference measure, shown in Figure 1(bottom). Here values seem largely indifferent to the sampling, except for the smallest sample of just over 3% of the whole collection. Again, the optimal value of λ is in the high end of the scale.

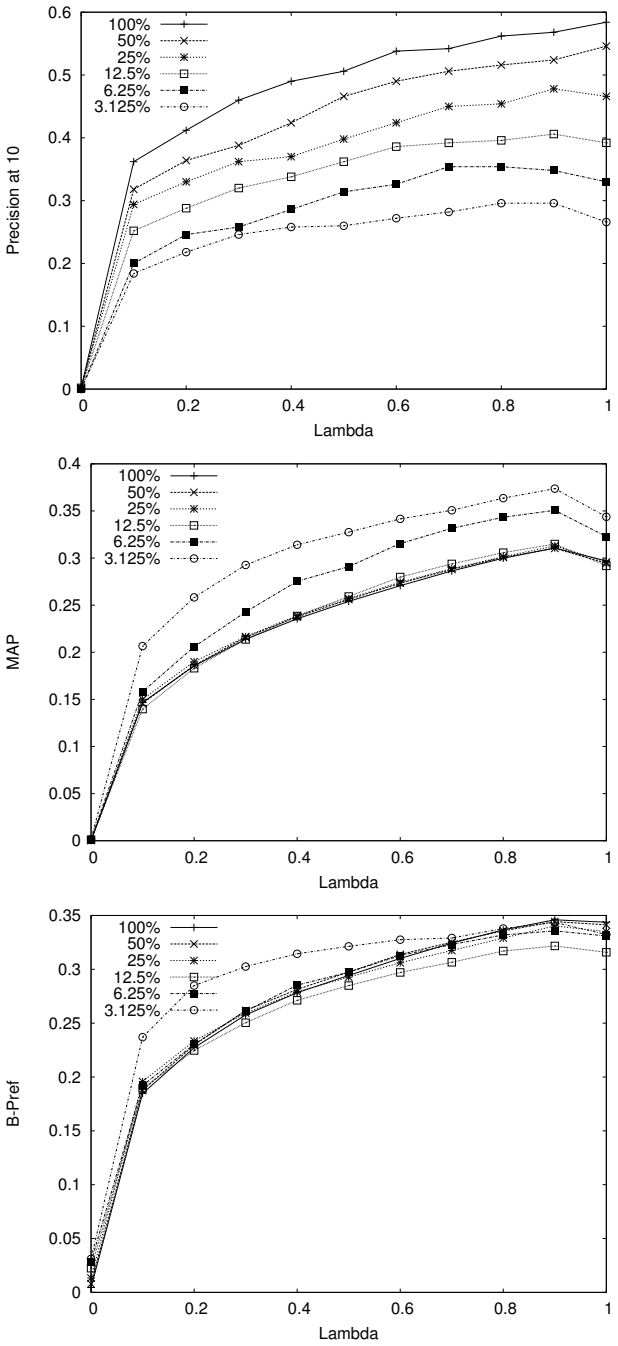

Figure 1: Scores over samples of the collection: (top) precision at 10; (middle) mean average precision; and (bottom) binary preference.

The results of our experiment with down-sampling the collection do not show a reversal in the amount of smoothing leading to optimal performance. Just as for the whole collection, smaller samples tend to favor high values of the smoothing parameter. There are two important qualification to make by the sampling strategy employed in this section. Firstly, we used only a single sample, rather than averaged over a large number of samples. Even though we took random samples, this may introduce accidental features especially when samples are small. Secondly, we only sampled a fixed retrieval run, hence the (relative) retrieval scores were determined by the collection as a whole. That is, the statistics from which the scores are estimated, and in particular the data-sparseness that smoothing methods address, do not change by the sampling method we applied in this section. Hence, a more thorough investigation is needed to study the effect of collection size on the amount of smoothing.

5 Conclusions

Our participation in the Terabyte track was inspired by a number of aims related to the size of the Terabyte track collection, we now draw some initial conclusions.

Our retrieval system did scale up to the 25 million documents in the GOV2 collection. Performance at query time is impressive, especially for the optimized implementation of the vector space model. With respect to indexing time, with over two week to build a full-text index we seem to have reached the limits of building a non-distributed index.

For the adhoc task, we saw that standard IR techniques on a full-text index lead to good performance. We found that, on the 2005 topics, the vector space model outperformed the language model, although the language model can be substantially improved by using less smoothing.

For the named page finding task, we found that, on the one hand, indegree and URL priors did not promote retrieval effectiveness, but that, on the other hand, the combination of different document representations improved retrieval effectiveness.

Last, but not least, we zoomed in on the role of smoothing and found that less smoothing leads to the best performance. Whereas this is roughly according to expectation for the named page finding topics, it is unexpected for the adhoc topics. In addition to known relations between retrieval effectiveness and collection size [1], there also seems to be a relationship between collection size and the appropriate amount of smoothing in the language modeling framework.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 612.066.302 and 640.001.501.

References

[1] D. Hawking and S. Robertson. On collection size and retrieval effectiveness. *Information Retrieval*, 6:99–150, 2003.

[2] ILPS. The ILPS extension of the Lucene search engine, 2006. http://ilps.science.uva.nl/Resources/.

[3] J. Kamps. Web-centric language models. In *Proceedings of the Fourteenth ACM Conference on Information and Knowledge Management (CIKM 2005)*. ACM Press, New York NY, USA, 2005.

[4] J. Kamps, G. Mishne, and M. de Rijke. Language models for searching in Web corpora. In E. M. Voorhees and L. P. Buckland, editors, *The Thirteenth Text REtrieval Conference (TREC 2004)*. National Institute of Standards and Technology. NIST Special Publication 500-261, 2005.

[5] J. Kamps, C. Monz, M. de Rijke, and B. Sigurbjörnsson. Approaches to robust and web retrieval. In E. M. Voorhees and L. P. Buckland, editors, *The Twelfth Text REtrieval Conference (TREC 2003)*, pages 594–599. National Institute of Standards and Technology. NIST Special Publication 500-255, 2004.

[6] Lucene. The Lucene search engine, 2006. http://lucene.apache.org/.

[7] Snowball. Stemming algorithms for use in information retrieval, 2006. http://www.snowball.tartarus.org/.

[8] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 49–56, Tampere, Finland, 2001. ACM Press.