Effects of Spam Removal on Search Engine Efficiency and Effectiveness

Matt Crane
Department of Computer Science
University of Otago
Dunedin, New Zealand
mcrane@cs.otago.ac.nz

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

ABSTRACT

Spam has long been identified as a problem that web search engines are required to deal with. Large collection sizes are also an increasing issue for institutions that do not have the necessary resources to process them in their entirety. In this paper we investigate the effect that withholding documents identified as spam has on the resources required to process large collections. We also investigate the resulting search effectiveness and efficiency when different amounts of spam are withheld. We find that by removing spam at indexing time we are able to decrease the index size without affecting the indexing throughput, and are able to improve search precision for some thresholds.

Categories and Subject Descriptors

H.3.1 [**Information Search and Retrieval**]: Content Analysis and Indexing – Indexing methods; H.3.3 [**Information Search and Retrieval**]: Information Filtering

Keywords

Information Retrieval, Web Documents, Spam, Procrastination

1. INTRODUCTION

Spam has been a long identified problem that web search engines must address. In 2009 TREC adopted the ClueWeb09 collection, a crawl of 1 billion web pages, as a standard collection for web track tasks. Some TREC submissions also made use of proprietary spam filters in their submissions [11].

Zuccon *et al.* [16] investigated the effect of withholding documents identified as spam on indexing and retrieval performance. They presented some interesting results, including a u-shaped relationship between amount of spam withheld from the index and the indexing time. They also show that there is no effect on retrieval time when spam was removed.

We aim to reproduce and explain these results, and to extend them by performing our own experiments of the effectiveness and efficiency that withholding spam documents has on a search engine.

We find that we are unable to replicate some of their results, but present some plausible reasons for them. Specifically, we find that the indexing time decreases consistently when more documents are excluded, and that retrieval time is strongly correlated with the size of the index that is generated.

ADCS'12, December 5-6, 2012, Dunedin, New Zealand

2. RELATED WORK

Cormack *et al.* [8] provide an in-depth examination of the effects of spam on the performance of the runs submitted to TREC 2009. They generated four different rankings of the spamminess of pages within the English ClueWeb09 dataset:

- **UK2006:** A set of labels trained against a small set of web pages containing 746 spam pages and 7,474 non-spam pages.
- **Britney:** Derived from results returned for popular queries given to commercial search engines.
- **Group X:** Manually labelled from results for queries from the 2009 TREC Ad-hoc task.
- **Fusion:** A combination of the other three methods.

Using these filters Cormack *et al.* were able to improve almost all runs that were submitted to TREC 2009. Two methods for modifying the submitted runs were consequently proposed, both of which were seen to improve performance. The first was to discard documents from the result set that did not meet a minimum threshold for non-spamminess, the second was to re-rank results using the spam score as a feature of the document.

The spam scores generated by Cormack *et al.* have subsequently been made available for use by other researchers, and have subsequently been used in a number of the top ranking systems in both the TREC 2010 and 2011, Web tracks for both the Ad-hoc and Diversity tasks as a threshold for indexing [1, 10, 13], a threshold for post-processing of returned results [9], and a feature to be used in document ranking [2, 12].

In 2010 TREC ran a spam identification track [5], for which the Fusion ranking generated by Cormack *et al.* was used a baseline. This baseline was not bettered.

Year	Number of Queries	TREC Query Numbers	Mean Query Length
2009	50	1–50	2.1
2010	48	51-99	2.02
2011	50	101–150	3.4

Table 1: Statistics for query sets being used.

Collection	ClueWeb09 Category B
Documents	50,220,423
Size	1.5TB
Unique Terms	96,298,556
Total Terms	75,614,656,698
Mean Document Length	1,505.66

Table 2: Collection statistics for ClueWeb09 Category B.

Zuccon *et al.* [16] investigated the effect that removal of spam had on the resources required to index ClueWeb09 Category B — the first 50 million English documents — and the effectiveness of the resulting indexes.

As Zuccon *et al.* stands, to our knowledge, as the only systematic investigation of the effects that spam removal at indexing time has on indexing performance and subsequent retrieval performance we seek to replicate and extend their experiments with the aim of applying the results learned on Category B to the Category A collection.

Intuitively removing documents from the indexing process will result in both a smaller index, and less time required to index. This was the opposite of the results presented in Zuccon *et al.*, who saw an increase in indexing time when removing a large proportion of the collection. Removing documents from the indexing process will also have an effect on the retrieval performance of the system, however, Zuccon *et al.* found no change in the retrieval time for a selection of ranking functions.

For our experiments we use the ATIRE search engine [14][1]. The experiments conducted in this paper are all performed on a machine with a quad cpu AMD Opteron 6276 2.3GHz 16-core, 512GB PC12800 memory, 6× 600GB 10000 RPM hard drives, and running Linux with kernel version 2.6.32.

Table 2 shows statistics of ClueWeb09 Category B. Table 1 shows some of the statistics for the query sets, we excluded queries 95 and 100 from the 2010 query set because no relevance judgements are available for them.

3. INDEXING

Zuccon *et al.* [16] propose an algorithm for modifying the indexing process to consider the spam score of the document being indexed, and only index those documents where its score met a given threshold. A list of documents to exclude is constructed prior to indexing, and this is consulted to determine whether to index a document.

Their results show that the indexing time forms a *u*-shape with respect to the given threshold, indicating that a higher threshold would take longer to index than a lower threshold.

One of the driving design decisions behind the ATIRE search engine [14] is the absence of any preprocessing. To keep aligned with this design decision, the spam filtering we added to ATIRE generates an internal list of documents to exclude at run-time from the complete list of <document, score> pairs. We, like Zuccon *et al.*, use the Cormack *et al.*'s Fusion scores.

However, we additionally take advantage of the fact that the spam scores are percentile scores. This allows us to construction an inclusion list if the threshold is greater than 50.

For instance, if given a threshold of 70, on the Category A collection 352,732,667 documents would be excluded from the index. Doing a binary search on this list of docids would require 29 string comparisons to identify whether a document should be excluded, compared with 27 when searching the list of documents that should be included. These comparisons are done on every document that is encountered during indexing, saving a total of over 1 billion string comparisons when indexing Category A.

Figure 1 shows the total indexing time for different thresholds using both the online and offline generation of lists. The figure shows the results across three runs for each method, with the median runs being joined and the slower and faster runs shown as error bars.

In an attempt to replicate the results from Zuccon *et al.* the offline method calculates only lists of documents to exclude. However, instead of the *u*-shaped results, we instead see a consistent drop-off in indexing times using both methods.

The reasons for the *u*-shape as seen in Zuccon *et al.* [16] is unclear and we were unable to reproduce it. They suggest that it "... may be caused by the procedure we used for loading the file containing the list of documents which do not have to be considered ...".

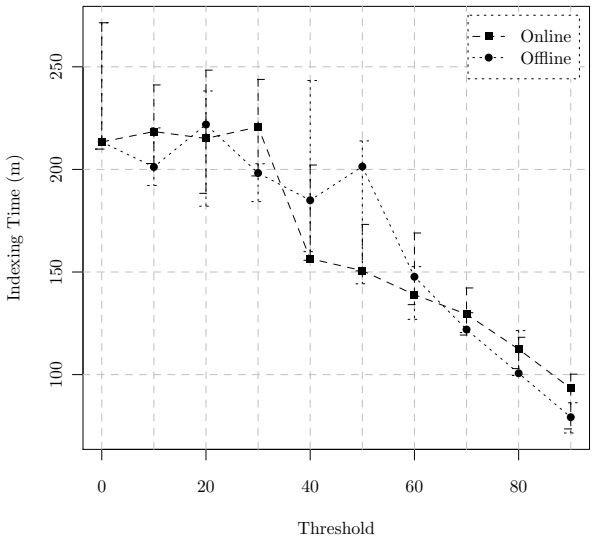

Figure 1: Comparative indexing time between online and offline calculation of documents to include or exclude from indexing with respect to the spam threshold given.

[1]Changeset: 56591fcef100

Threshold	Documents	Unique Terms	Total Terms	Mean Document Length
0	50,220,423	96,298,556	75,614,656,698	1,505.66
10	48,736,112	74,805,408	72,664,843,957	1,490.99
20	46,432,700	69,693,504	69,021,947,305	1,486.49
30	43,718,178	64,173,727	64,714,615,144	1,480.27
40	40,844,719	58,008,256	60,003,421,198	1,469.06
50	37,655,996	51,750,917	54,792,100,355	1,455.07
60	33,836,981	45,082,561	48,802,927,422	1,442.30
70	29,038,220	37,621,793	41,525,104,236	1,430.02
80	23,148,047	29,487,702	32,832,607,822	1,418.37
90	15,374,591	19,745,587	21,417,223,927	1,393.03

Table 3: Index statistics for indexes generated with different threshold values.

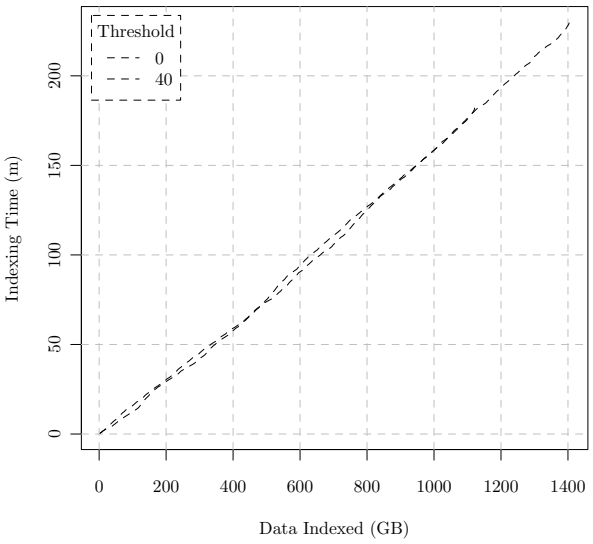

Figure 2: Indexing throughput for two different thresholds of spam removal as measured by time taken relative to data indexed.

We believe this is a reasonable explanation as Zuccon *et al.* use C++'s `>>` operator to read documents into a `std::map`, while the spam filter added to ATIRE performs a block read of the file, and performs a linear scan to set up pointers with no unnecessary copying of data.

We also make note of the large difference in total indexing times (≈1750 minutes compared with 210) between the two indexing processes when performing no spam filtering. This suggests that there may also be an underlying engineering component to the discrepancy in results.

We see only a marginal overhead in the online calculation of the lists of docids to discard or keep, with the difference between median runs with a threshold of 70 being 7 minutes, or ≈6%.

Figure 2 shows the time taken to index as a function of the amount of data that has been indexed (without spam removal, and with a threshold of 40). This figure shows that by withholding spam from the index, we do not substantially affect the throughput rate of the indexing system, which indicates that the time taken to determine whether to index a document is negligible when compared to the time needed to index that document. Further evidence that the u-shape is due to factors outside of the spam identification itself.

We also note from this graph the near linear relationship between indexing time and data indexed (r^2 value of 0.9985).

Figure 3 shows the resulting index sizes for each of the indexes generated. We see a similar drop off in relative index sizes as Zuccon *et al.* [16], with a threshold of 40 generating an index that is 24GB in size, while their index is ≈135GB for a threshold of 45.

Table 3 show some statistics — number of documents, number of unique terms, number of total terms, and average document length — of the generated indexes for each different threshold, for example at a threshold of 40 we index 41 million documents, containing 60 billion instances of 58 million unique terms and an average document length of 1,469 terms.

Interestingly, we see a sharper drop off in unique terms than total terms, as well as a consistent decline in the average document length. This indicates that documents that are identified as spam have a higher proportion of unique terms, and tend to be longer.

The ATIRE search engine defaults to Variable Byte compression. The ATIRE search engine also stores the postings lists using impact ordering on term frequency, which is itself a form of compression [14].

We note that the number of documents remaining in the index does not match the thresholds given. This is because the spam scores were generated across the complete set of 500 million English documents of ClueWeb09 Category A. Category B contains the first documents crawled, and due to the nature of the crawl contains documents that are less likely to be spam. If one wanted to remove half the documents in Category B, then a threshold in the 70s should be specified.

4. SEARCHING

4.1 Effectiveness

Having now investigated the performance of indexing under different thresholds for spam removal, we now investigate the efficacy of the search engine across these different indexes. We measure search performance against the qrels from the diversity task for the 2009 queries to enable training on these queries and testing on the 2010 and 2011 queries.

As a precision measure we use ERR-IA as described by Chapelle *et al.* [4] as the primary measure as it is used by TREC. We also report α-nDCG [7] scores to enable compar-

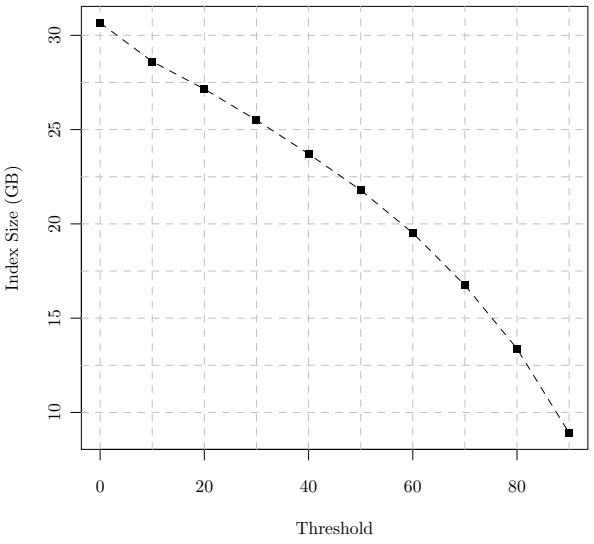

Figure 3: Size of generated index for different threshold values.

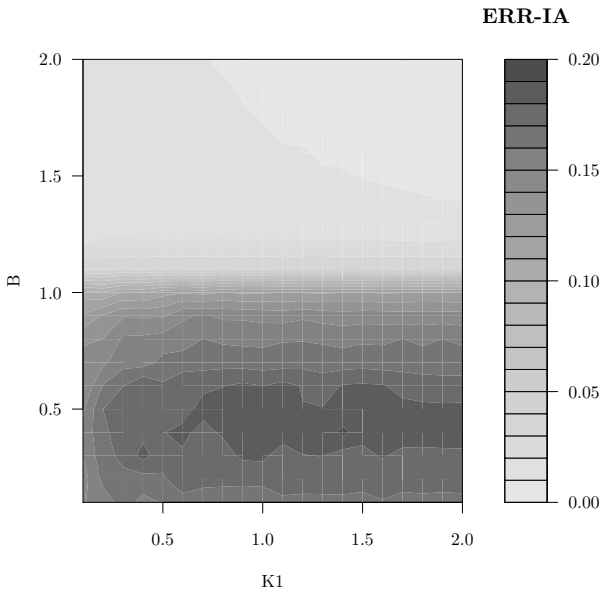

Figure 4: Example surface generated from the grid search of BM25 parameters on index with spam threshold set to 40. A darker shade indicates a higher ERR-IA@20.

ison with prior reported results.

For each index we perform a grid search to find the best parameters for the BM25 function. The grid search is performed across the 50 queries from the 2009 web track at TREC, with ERR-IA@20 as the primary performance measure.

Figure 4 shows an example surface generated by the grid search performed on the index with spam threshold set to 40. The darker the shading, the higher the ERR-IA@20, with a peak value of 0.1931 when k1=1.4 and b=0.4. The other, traditional, variables within the BM25 ranking functions are ignored due to the modified version implemented within the ATIRE search engine [14].

Zuccon *et al.* [16] identified a threshold of 70 provided the best results when considering both ERR-IA@10 and α-nDCG@10 across a range of ranking functions. Cormack *et al.* [8] identified a threshold of 50 for Category B when altering runs submitted to TREC. Both identify an upside down u-shape, with no filtering performing the worst.

Figure 5 shows the results from the grid searches as a function of the spam threshold specified during indexing. These are optimal scores with our ranking function. We identify the same upside down u-shape, although we find that the performance of no spam filtering to perform better than a threshold of 90. When targeting ERR-IA@20 we obtain a peak value of 0.1931 when a threshold of 40 is given at indexing time.

Figure 6 shows the results of evaluating using α-nDCG@20 when using the BM25 parameters found from the grid search against ERR-IA@20. We find the same upside down u-shape, suggesting that this is evaluation function-independent. Interestingly, we find that no spam filtering performs better than specifying a threshold of 70 or higher. A peak α-nDCG@20 value of 0.3237 is obtained with a threshold of 30. These are optimal scores using our ranking

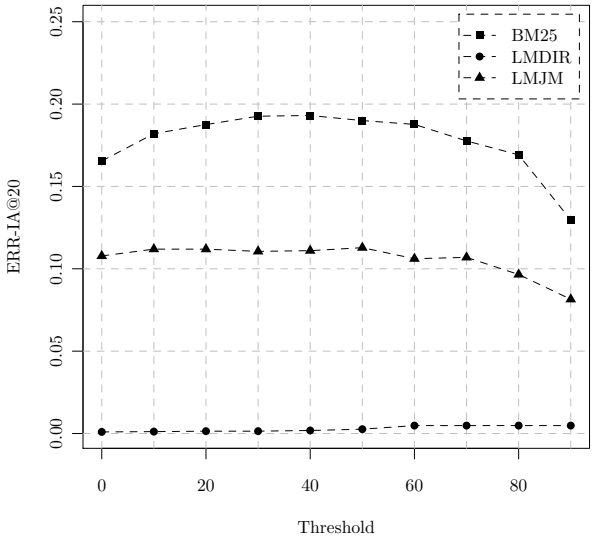

Figure 5: Best results from grid search of BM25 values for different thresholds targeting ERR-IA@20 as the evaluation function.

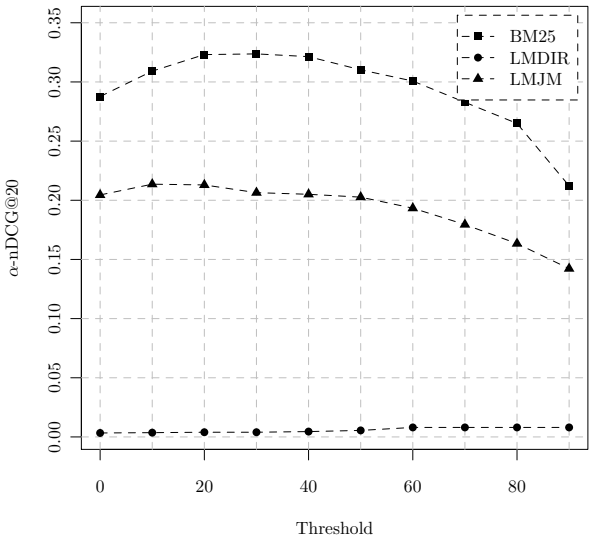

Figure 6: Best results from grid search of BM25 values for different thresholds using α-nDCG@20 as the evaluation function from search against ERR-IA@20.

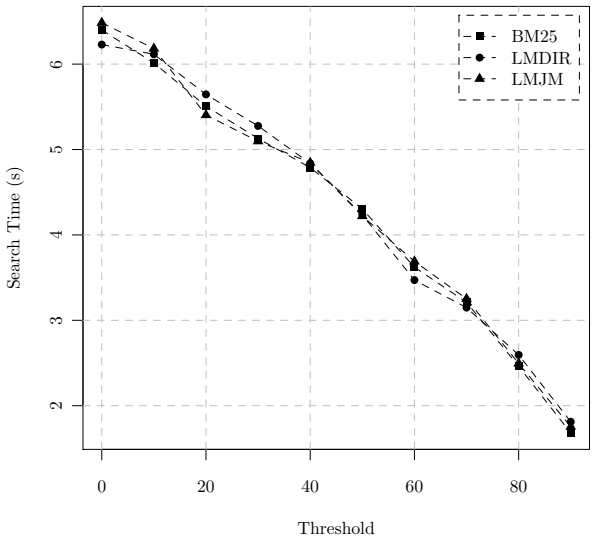

Figure 7: Average time taken to search with respect to spam removal threshold.

function.

Because of its use as the primary ranking function in the TREC diversity tasks, the rest of the paper focuses on ERR-IA@20, but will reports α-nDCG values as appropriate.

4.2 Efficiency

Zuccon *et al.* [16] identify an interesting phenomenon when evaluating the time to search, where only the LMJM [15] ranking function (Unigram Language Model with Jelineker-Mercer smoothing) showed any change in query throughput. They suggest that this change in throughput is caused by implementations in Indri. Intuitively, however, a smaller index *should* result in higher throughput, with time taken to search 0 documents being near 0.

Figure 7 shows the retrieval time per query for indexes pruned of spam (at various thresholds). We see an almost linear drop off in relation to the spam threshold regardless of ranking function. For instance removing no spam results in an average search time of 6 seconds, while a threshold of 90 results in an average search being performed in a quarter of the time.

Times for BM25 were averaged across 400 runs of the 50 queries used during the grid search, while LMJM and LMDIR [15] (Unigram Language Model with Dirichlet smoothing) times were averaged across 3 runs of the same queries. For LMJM and LMDIR, we utilised the same parameters as Zuccon *et al.*, that is, $\mu = 3000$ for LMDIR, and $\lambda = 0.01$ for LMJM.

We cannot suggest a reason for no change in retrieval performance identified by Zuccon *et al.*, other than their own, specific implementations in Indri.

5. FURTHER REDUCTIONS

Having generated an index that provides good performance on the 2009 queries, we now question whether we can further reduce the size of the index without compromising on the precision of the results.

We can further reduce the size of the index by stopping terms. The selection of these terms can influence the performance of the search engine, by removing potential query terms, and by altering the statistics of the documents that are indexed. We select numbers and HTML tags as obvious candidates for removal.

By stopping numbers we are able to reduce the size of the index by 3GB, by stopping tags we can reduce the size of the index by 1GB. Stopping numbers reduces ERR-IA@20 to 0.1827. Whereas stopping the tags has no effect on retrieval performance, because ATIRE does not consider the presence of tags in the document or collection statistics.

The terms that are of the most importance when considering a document's relevance to a query are the "middle" terms, that is, those terms that are not infrequent, and also not frequent. To this end, we stop those terms with a document frequency of 1, reducing the index size by 1GB and resulting in an ERR-IA@20 of 0.1931.

When all of the stopping conditions are selected, the resulting index is reduced in size by 20%. This is slightly less than the sum of the individual improvements because some terms may be stopped under multiple conditions. This reduction in index size is accompanied by an approximate 15% decrease in time required to search.

Unfortunately, the ERR-IA@20 for the stopped index drops from 0.1931 to 0.1826, which is a statistically significant change (p-value < 0.01 on a two-tailed pairwise t-test). We consider this change in performance to be acceptable given the increase in efficiency. This efficiency versus effectiveness trade-off has been explored by others [3].

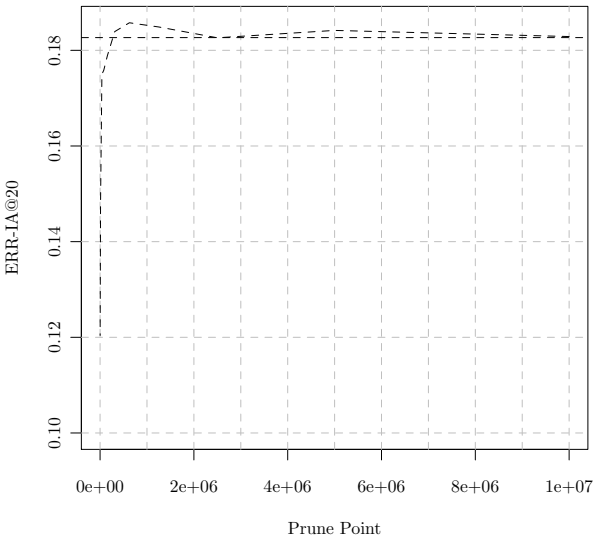

Figure 8: Effect of static pruning on ERR-IA@20 on index with numbers, tags and infrequent terms removed and spam threshold set to 40.

	ERR-IA		α-nDCG	
Year	@10	@20	@10	@ 20
2009	0.1792	0.1842	0.2892	0.3105
2010	0.1863	0.1995	0.2533	0.3029
2011	0.4035	0.4123	0.4803	0.5120

Table 4: Evaluation scores on the different query sets using BM25 learned on 2009 queries, with spam threshold set to 40.

We can further reduce the size of the index by statically pruning the postings lists. That is, only allowing at most the first n documents of each postings list for a term to remain in the index. Recall that our index is impact ordered on term frequency and so the first documents have the highest tf scores. The ATIRE search engine support this at both search-time and index-time parameters, allowing us to train at search time and to re-index with the optimal value.

Figure 8 shows the ERR-IA@20 as a function of this pruning value, with the stopped, unpruned performance shown as the black dashed line. We see that it takes substantial pruning before the retrieval performance is degraded, and some pruning actually increases the performance. A pruning value of 300,000 provides an ERR-IA@20 of 0.1840 with a p-value of 0.051 (not significant) when compared to the unstopped, unpruned index.

When pushing this static pruning to indexing time, we reduce the size of the index by only 400MB. This relatively small change in the index size is accompanied by a much larger relative reduction of search time from 5 seconds, to 850 milliseconds, when our index is stored on disk. This is because the postings lists do not take much space inside the index, but are processed exhaustively during search.

The results for all query sets on this final generated index

	ERR-IA		α-nDCG	
Year	@10	@20	@10	@ 20
2009	0.1044	0.1105	0.1732	0.2021
2010	0.1048	0.1124	0.1702	0.1968
2011	0.2934	0.2995	0.3667	0.3883

Table 5: Evaluation scores on the different query sets using LMJM, $\lambda = 0.01$, with spam threshold set to 40.

	ERR-IA		α-nDCG	
Year	@10	@20	@10	@ 20
2009	0.0026	0.0026	0.0056	0.0055
2010	0.0004	0.0014	0.0017	0.0060
2011	0.0000	0.0000	0.0000	0.0000

Table 6: Evaluation scores on the different query sets using LMDIR, $\mu = 3000$, with spam threshold set to 40.

are shown in Table 4. The scores for 2009 should not be used in direct comparison as these were the queries used for training, they are included for completeness. On the 2009 queries, the final index has a p-value of 0.04 when compared to the index generated with no filtering applied.

We note that the results on the 2011 queries would have placed as 6^{th} in the diversity task at TREC [6] without performing any explicit diversification and using just BM25 for ranking. We do, however, concede that this is not equivalent to submitting a run to TREC.

Table 7 shows the results of applying the same spam thresholding, stop word removal and static pruning on the Category A collection. The results for a threshold of 70 are also shown, as this has been shown in previous work to provide the best threshold for this collection [8]. The parameters for BM25 were again trained on the 2009 query set by grid search against ERR-IA@20. The thresholds of 40 & 70 yield index sizes of 120GB and 62GB and allow searching in an average time of 4.5 and 1.8 seconds respectively.

6. FUTURE WORK

We can further reduce the overhead of the search to find if a document should be excluded at indexing time. The files within each of the `.warc.gz` files are stored in sequential docid order. By performing a binary search when we inspect the first document within the archive, and then using a linear scan for the rest of the archive we hypothesise that we can drastically reduce the number of string comparisons required.

The index size can also be further decreased by utilising stemming at indexing time. This would have the effect of conflating terms together and reducing the size of the dictionary. However, this would require re-learning BM25 weights, a good static pruning value, and which terms could still be effectively stopped. The inclusion of more items to consider at indexing time quickly turns this process into a multi-objective optimisation problem.

In exploring the effect of the spam score on search effectiveness, Cormack *et al.* [8] found that by re-ranking results by incorporating the spamminess score improved precision. Indeed this approach has been taken by a number of top ranking runs in the TREC 2011 web track tasks [2, 12]. We

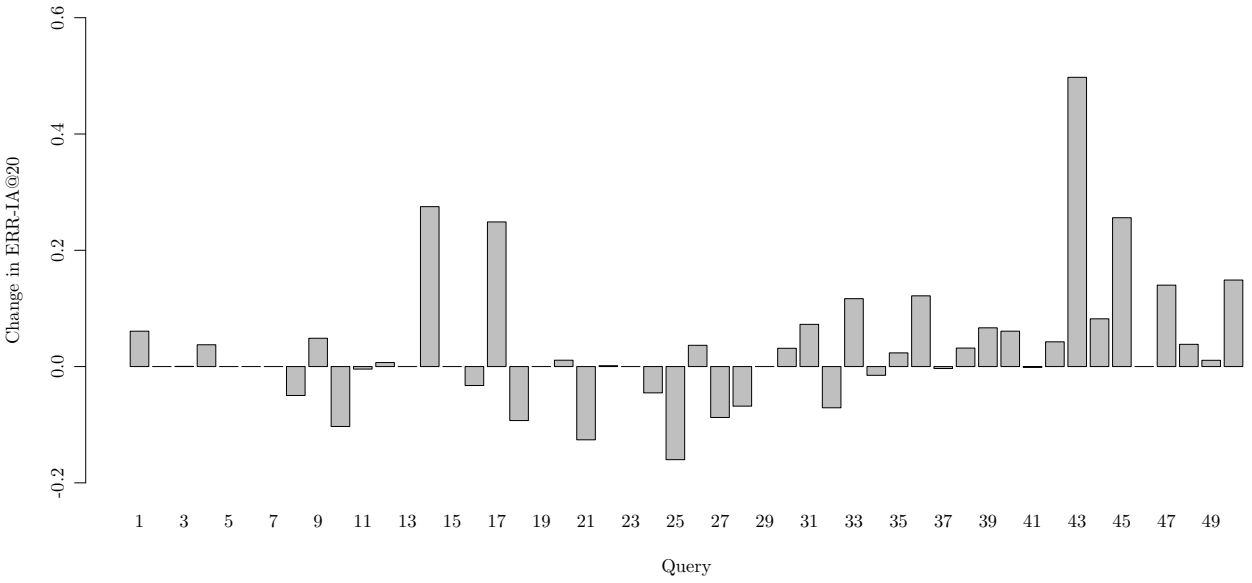

Figure 9: Change in ERR-IA@20 for individual queries in the 2009 query set from no filtering to final index. A positive value means an improvement for that query, while a negative value indicates performance degradation.

		ERR-IA		α-nDCG	
Year	Threshold	@10	@20	@10	@ 20
2009	40	0.1096	0.1178	0.1781	0.2069
	70	0.1217	0.1289	0.1902	0.2204
2010	40	0.1430	0.1509	0.1991	0.2289
	70	0.1778	0.1883	0.2384	0.2794
2011	40	0.3595	0.3718	0.4446	0.4878
	70	0.4067	0.4173	0.4846	0.5216

Table 7: Evaluation scores on Category A between threshold learned on Category B with previous research, with learned stop word removal and static pruning.

will explore this alternative approach to the spam problem.

The imminent release of ClueWeb12 provides an opportunity to examine the effects that spam removal will have on that corpus.

7. DISCUSSION & CONCLUSION

We have investigated the effect of removing spam documents on both indexing and retrieval performance. We found a consistent decrease in time to index as the amount of spam removed during indexing increased, and this was accompanied by a consistent decrease in index size. We also showed that by removing spam from the indexing process we did not alter the near linearity of the indexing system.

For each generated index, we then tuned our implementation of BM25 and identified the threshold that provided the best effectiveness when measuring ERR-IA@20 on the TREC 2009 web track queries. We then investigated the efficiency of searching these indexes, and found a clear relationship between index size and search throughput.

These results are in contrast to those found within Zuccon *et al.* [16], which suggests their observed behaviour was extra to the process of indexing and searching (e.g. reading the spam list).

We then further reduced the index size without decreasing effectiveness by introducing stopping and static pruning. We found that while stopping numbers, tags and terms with a document frequency of one made a statistically significant decrease in effectiveness, static pruning at 300,000 improved performance to a statistically insignificant level when compared to the unstopped, unpruned index.

The final index generated for Category B is 20GB, and allows searching to be carried out in 400 milliseconds when the index is loaded entirely into memory, averaged across 3 runs of the 2009 TREC queries.

The retrieval effectiveness of the final index is comparable to the top-ranking systems for the diversity task at TREC 2010 and 2011 despite only using BM25 without anchor text or PageRank scores.

Figure 9 shows the change in ERR-IA@20 for individual queries in the 2009 query set from performing no filtering to the final index as discussed in this paper. A positive value indicates that performance for that query improved, while a negative value indicates that the performance for that query was degraded.

We see that there are a few queries that are improved substantially, most notably query 43 — "the secret garden" — which was improved substantially. We also see a small reduction in the performance of a selection of queries that at cursory glance would suggest should be improved by spam filtering — 10: "cheap internet", 21: "volvo", 32: "website design hosting".

8. REFERENCES

[1] M. Bendersky, D. Fisher, and W. Croft. Umass at trec 2010 web track: Term dependence, spam filtering and quality bias. In *Proceedings of the Text REtrieval Conference (TREC 2010)*, 2010.

[2] B. Billerbeck, N. Craswell, D. Fetterly, and M. Najork. Microsoft Research at TREC 2011 Web Track. In *Proceedings of the Text REtrieval Conference (TREC 2011)*, 2011.

[3] S. Büttcher and C. Clarke. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proceedings of the Text REtrieval Conference (TREC 2005)*, 2005.

[4] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630. ACM, 2009.

[5] C. Clarke, N. Craswell, I. Soboroff, and G. Cormack. Overview of the TREC 2010 Web track. In *Proceedings of the Text REtrieval Conference (TREC 2010)*, 2010.

[6] C. Clarke, N. Craswell, I. Soboroff, and E. Vorhees. Overview of the TREC 2011 Web track. In *Proceedings of the Text REtrieval Conference (TREC 2011)*, 2011.

[7] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666. ACM, 2008.

[8] G. Cormack, M. Smucker, and C. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information retrieval*, 14(5):441–465, 2011.

[9] C. Hauff and D. Hiemstra. University of Twente @ TREC 2009: Indexing half a billion web pages. In *Proceedings of the Text REtrieval Conference (TREC 2009)*, 2009.

[10] J. Kamps, R. Kaptein, and M. Koolen. Using anchor text, spam filtering and wikipedia for web search and entity ranking. In *Proceedings of the Text REtrieval Conference (TREC 2010)*, 2010.

[11] J. Lin, D. Metzler, T. Elsayed, and L. Wang. Of Ivory and Smurfs: Loxodontan MapReduce experiments for web search. 2009.

[12] R. McCreadie, C. Macdonald, R. Santos, and I. Ounis. University of Glasgow at TREC 2011: Experiments with Terrier in Crowdsourcing, Microblog, and Web Tracks. In *Proceedings of the Text REtrieval Conference (TREC 2011)*, 2011.

[13] M. Smucker. Crowdsourcing with a crowd of one and other TREC 2011 crowdsourcing and web track experiments. In *Proceedings of the Text REtrieval Conference (TREC 2011)*, 2012.

[14] A. Trotman, X. Jia, and M. Crane. Towards an efficient and effective search engine. In *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 40–47, 2012.

[15] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

[16] G. Zuccon, A. Nguyen, T. Leelanupab, and L. Azzopardi. Indexing without spam. In *Proceedings of the 16th Australasian Document Computing Symposium (ADCS 2011)*, pages 6–13, 2011.