Indexing without Spam

Guido Zuccon
The Australian e-Health Research Centre
CSIRO
QLD, Australia
Guido.Zuccon@csiro.au

Anthony Nguyen
The Australian e-Health Research Centre
CSIRO
QLD, Australia
Anthony.Nguyen@csiro.au

Teerapong Leelanupab
Faculty of Information Technology
KMITL
Thailand
t.leelanupab@it.kmitl.ac.th

Leif Azzopardi
School of Computing Science
University of Glasgow
Scotland, UK
Leif.Azzopardi@glasgow.ac.uk

Abstract

The presence of spam in a document ranking is a major issue for Web search engines. Common approaches that cope with spam remove from the document rankings those pages that are likely to contain spam. These approaches are implemented as post-retrieval processes, that filter out spam pages only *after* documents have been retrieved with respect to a user's query. In this paper we propose removing spam pages at indexing time, therefore obtaining a pruned index that is virtually "spam-free". We investigate the benefits of this approach from three points of view: indexing time, index size, and retrieval performance. Not surprisingly, we found that the strategy decreases both the time required by the indexing process and the space required for storing the index. Surprisingly instead, we found that by considering a spam-pruned version of a collection's index, no difference in retrieval performance is found when compared to that obtained by traditional post-retrieval spam filtering approaches.

Keywords Information Retrieval; Index Pruning; Spam; Web search; Efficiency.

1 Introduction

The presence of spam in a document ranking is a major issue for Web search engines. For example, in TREC 2009 Web Track, which is based on a large crawl of the Web (i.e. the ClueWeb collection, about 1 billion web pages), it has been noted that spam harmed performance of retrieval systems [10, 11]. To cope with this problem, participants to the TREC Web Track have implemented strategies of post-retrieval processing that filter out pages deemed as spam [9, 12]. Similarly, Cormack et al. have produced four spam lists for the ClueWeb collection. These consist of a label associated to each document indicating the likelihood of the document being spam [8]. They also showed that by removing spam employing a post-retrieval process substantially improves retrieval performance of TREC 2009 systems.

In this paper we tackle the problem of spam in Web collections (specifically the ClueWeb collection) from a different perspective. While previous approaches filtered out spam from the retrieval results using a post-retrieval process, we modify traditional indexing procedures, by introducing a spam filtering step at indexing time. By doing so, web pages that are deemed to be spam with respect to some threshold level are not indexed, and therefore, neither are they retrieved. By not considering spam documents at indexing time, modified index statistics are obtained, if compared to indexes obtained using traditional indexing procedures. Our strategy acts as an index pruning technique, driven by the spam scores of documents. We expect our technique to benefit by the common characteristics of indexing pruning strategies: reduced index size and reduced indexing time. However, index pruning techniques have been shown to degrade retrieval performance of systems. But, does pruning with respect to spam documents harm retrieval performance as well?

To investigate benefits and drawbacks of *"indexing without spam"*, we conducted a number of retrieval experiments across several traditional retrieval models on the TREC ClueWeb 2009-2010 Web Track dataset. We found that the use of indexes used through our indexing without spam procedure does not harm retrieval performance. Instead, often this procedure offers better effectiveness than traditional indexing with or without post-retrieval removal of spam documents, while reducing indexing and retrieval time.

The paper continues as follows. In section 2 we outline the method of indexing without spam. Subsequently we present related works in the area of in-

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

dex pruning and spam identification and removal in the context of the ClueWeb collection and the TREC Web Track (section 3). Our research questions are stated in section 4, while section 5 presents and discusses our experimental settings and results. Finally, the paper concludes in section 6.

2 Indexing without Spam

Assume that a spam list for a corpus is given, where a score is associated to each document indicating its likelihood of being spam. For ease of exposition, let assume that a low spam score indicates that a document is very likely to be spam, with 0 being associated with the documents that are most probable spam. Similarly, a high score would suggest that a document is unlikely to be spam, with 99 indicating documents that are least probable to be spam.

Given these settings, we propose to modify the traditional indexing processes, such that both corpus and spam list are provided as input to the indexer. Thereafter, the indexer is instructed to index only those documents in the corpus for which their spam list scores are higher than a threshold th. Algorithm 1 outlines the pseudocode of such indexing procedure, where $\mathcal{SL}(d)$ returns the spam score associated with document d.

This approach resembles the notion of index pruning, where the inverted index is compressed by avoiding storing some of the statistical information associated to terms or documents.

Algorithm 1 Indexing without spam

Input: a corpus of documents $\mathcal{D}$; a spam list $\mathcal{SL}$ containing pairs of documents and spam scores; a threshold value th
Output: an index $\mathcal{I}$

```
for all d ∈ D do
  if SL(d) > th then
    index d in I
  else
    ignore d
  end if
end for
return I
```

3 Related work

3.1 Index Pruning

As the approach presented in section 2 is similar to pruning an index (lossy compression), we briefly revise and compare some index pruning strategies proposed in the literature. In particular, we focus on static index pruning strategies, as opposed to dynamic strategies which are applied at query time (for example [13]).

The strategy of posting pruning by sparsification of the index table, as proposed by Carmel et al. [2], consists of calculating the importance of individual postings in the index, and then removing from the index those postings that are less informative. This results in the removal of the associated terms from documents, thus varying the statistics associated to document lengths and term frequencies, and possibly lead to the complete removal of a term from the index. Therefore, this pruning strategy acts at the term level.

While the sparsification of the index table (at term level) does not necessarily imply the complete removal of a term from the index, the term pruning strategy of Blanco and Barreiro [1] prescribes that uninformative terms should be completely removed from the index. Therefore, such terms are treated similarly to stopwords. This approach radically modifies the statistics of the pruned index, affecting the document length statistics. Ultimately, the strategy might affect the presence of some documents, if these contained only terms that have been pruned.

While the previous approaches act at a term level, the strategy proposed by Zheng and Cox prunes the index at a document level [20]. A score is computed for each document in the collection, based on the entropy of the terms in the documents. Documents are thereafter selected or rejected for indexing depending upon their score being higher or lower than a set threshold. This solution affects the presence of documents, the average document length, and the inverse document frequency.

Common to index pruning strategies is that a trade-off is found between index size and retrieval performance: higher levels of index pruning (i.e. smaller indexes) translate in worse retrieval performance, with no-pruning obtaining usually the best retrieval performance.

The document pruning strategy of Zheng and Cox is the closest pruning strategy to that proposed in this paper. In contrast to Zheng and Cox, here we do not consider the entropy of terms within documents for deciding which documents should be excluded from the indexing process. Instead, we reject documents for indexing according to their scores with respect to spam features.

3.2 Spam Identification and Removal

The presence of spam among top ranked search results is a problem that harms the retrieval effectiveness of Web search engines. Participants to the TREC 2009 Web retrieval Track have noted the impact of spam on system effectiveness [10, 11], while others attempted to prune spam documents from the ranking through a post-retrieval process [9, 12]. In particular, Hauff and Hiemstra used a spam detection algorithm that relies on page content, page title features and URL form [9]. Once spam documents were identified, they were removed by the document ranking through a post-retrieval processing stage. Similarly, Lin et al. used Yahoo!'s proprietary adult, spam, and document quality classifiers to identify and remove spam once documents have been retrieved by their system [12].

While several methods have been proposed in the literature to identify spam documents (e.g. [3, 14]), we focus on the work of Cormack et al. [8]. This is because they have already successfully applied their spam detection and filtering methods to web retrieval (and in particular to the same dataset we employ in this study) and studied the effect of spam on retrieval performance. In particular, Cormack et al. adapted content-based email spam filters based on three different training sets to the task of identifying spam web pages in ClueWeb. The training sets were:

- UK2006: a (small) set of spam and non-spam labels for web pages, containing 767 spam pages and 7,474 non spam pages.
- Britney: a set of training examples heuristically built from popular search queries as reported by popular web search engines' statistics. Following this method, the top 10 ranked pages retrieved by the Indri retrieval system in answer to the popular queries were deemed as being spam. While, 10,000 documents belonging to the collection for which their URIs were found to be also present in the Open Directory Project[1] were classified as non-spam.
- Group X: this training set consists of 756 documents for which human assessments regarding the presence of spam were collected by Cormack et al.

Furthermore, the single scores obtained by the three filters can be combined together through a naïve Bayes combination: this approach (called *"Fusion"*) has been shown to be the most effective in the context of spam identification and document retrieval when using the ClueWeb dataset. In the experiments reported in section 5, we employ the spam list generated with the Fusion method, which is publicly available[2]. For each document in ClueWeb, the spam list contains a percentile score, which indicates the percentage of the documents in the corpus that are "spammier". For example, if a document received a percentile score of 95 it means that it belongs to the 5% of the documents that are least probable to be spam. While, if a low percentile score is associated to a document, then this is likely to be a spam page: a document with score 0 belongs to the "spammiest" 1% of the documents.

The research we have reviewed in this section provides us the motivations for studying the effect of spam on indexing, both in terms of indexing and retrieval performance obtained by employing a spam-pruned index.

4 Research Questions

Previous approaches that account for spam in document retrieval have tackled the problem at the *retrieval phase*: documents are retrieved in response to a query, those that are identified as possible spam are filtered out, and the remaining documents are returned to the user that initially issued the query. In this work we investigate an alternative solution, where documents that are identified as being spam are removed during the *indexing phase*. Specifically, when indexing a collection, we suggest that documents that may be considered to contain spam are filtered out and therefore not indexed at all. This approach leads to the construction of an index that does not contain spam documents (as identified by the spam identification algorithm that is employed). Furthermore, the resulting index statistics, such as term counts, idf-s, average document lengths, etc, are effectively affected by the absence of evidences that would normally have been drawn from documents considered as spam.

In this paper, we investigate the differences between the two approaches to document retrieval with spam removal. Specifically, we explore the following research questions:

RQ1 How does spam removal at indexing time affect the *indexing process*?

RQ2 How does spam removal at indexing time affect the *indexes* that are created? And, how different are these indexes from those created using the standard indexing procedure?

RQ3 How does spam removal at indexing time affect *document retrieval*? Do index statistics obtained by an index that has been pruned from spam at indexing stage lead to worse retrieval performance, if compared to that obtained using retrieval-time spam removal? Does spam removal at indexing time affect retrieval time as well?

5 Experiments

5.1 Experimental Methodology

To empirically investigate our research questions, we employed the ClueWeb 09 collection (we used Category B only in this initial investigation), consisting of more than 50 million web pages. We indexed documents using Indri 5.0 [17][3], after stop-words removal and stemming conflation with the Krovetz stemmer. In the retrieval experiments, we used the TREC 2009 and 2010 Web Track topics[4], and four standard retrieval models as implemented by Indri: Okapi BM25 [16], Unigram Language Model with Dirichlet smoothing (LMDIR), Unigram Language Model with Jelineker-Mercer smoothing (LMJM) [18], and Unigram Language Model with two stage smoothing (LM2S) [19]. For the retrieval methods based on language model, we used the parameter

[1] `http//rdf.dmoz.org`

[2] `http://durum0.uwaterloo.ca/clueweb09spam/`

[3] `http://lemurproject.org/indri.php`

[4] Specifically, we used all topics released in TREC 2009 Web Track and 48 topics released in TREC 2010 Web Track: topics 95 and 100 from TREC 2010 were excluded because no relevance assessments were provided for them.

values that obtained the best retrieval effectiveness in the experiments of Zhai and Lafferty on the TREC 8 Web collection with short queries [18]:

- LMDIR: $\mu = 3,000$
- LMJM: $\lambda = 0.01$
- LM2S: $\lambda = 0$, $\mu = 3,000$

For BM25 we used the standard settings suggested in [15], i.e.:

- BM25: $b = 0.5$, $k_1 = 2$, $k_2 = 0$, $k_3 = 8$

To identify spam, we employed the "Fusion" spam list created by Cormack et al. [8], where a percentile score is associated with each ClueWeb document indicating its level of spam, as described in section 3.2. Note that the percentile scores refer to the whole ClueWeb dataset, while here we consider category B only. As a result, in our experiments, the number of pages that have a percentile score of x or lower is not equivalent to the $x\%$ of pages in the category B dataset[5].

These common settings have been used to implement the following 3 systems:

System S_1: a traditional information retrieval system, where all the documents contained in the collection are indexed and when a query is issued retrieval is performed according to one of the standard models listed above.

System S_2: a modification of System S_1, where a post-retrieval process is activated to remove those documents that are identified in the spam list as containing spam with a percentile score smaller or equal to a threshold th. This system implements the *post-retrieval spam removal* approach.

System S_3: a modification of System S_1, where documents that are identified in the spam list as containing spam with a percentile score smaller or equal to a threshold th are not considered during the indexing process. This system implements our *indexing without spam* approach.

Systems S_1 and S_2 are based on the standard Indri 5.0 toolkit, where for System S_2 we implemented a post-retrieval filtering process that removes from the result rankings those documents that were flagged as spam. To implement System S_3, we modified the indexing process of Indri 5.0 toolkit, including a filtering stage to exclude spam documents before the system performs indexing. The spam list $\mathcal{SL}$ is read in memory using the C++ operator >>, and document identifiers that refer to those documents which do not have to be indexed are stored into a standard C++ `std::map<string,bool>` container. The storing operation (operator `[]`) is logarithmic in the size of `map`. At indexing time, once a document identifier is read by the indexing process, a `count` operation is performed on the `map` container: if the result of this operation returns a value greater than zero (i.e. the `map` contains the searched document) the document is not indexed, and the next document is processed. The `count` method requires logarithmic time in the size of the `map`.

In our experiments we empirically investigate four settings of the threshold parameter[6] th, i.e. $th = \{20, 45, 70, 95\}$. Moreover, we studied the performance of S_1, which corresponds to S_2 and S_3 when $th = 0$.

To assess the difference between the common indexing procedure and the approach based on indexing without spam, we record the time required for indexing by the standard Indri 5.0 toolkit and by our own modification of that software for all levels of the spam threshold th. Indexing and retrieval were performed on a high performance server, fitted with 16 Intel Xeon X7350 (2.93GHz) CPUs and 128 GB of RAM. For each setting of th, indexing and retrieval were performed separately, and no other user-process was in execution on the server when indexing the collection.

In our empirical study, indexing was performed only once for each level of th: thus the time we recorded is that of a single execution of the indexing process. A more accurate approach would be to consider a number of repetitions of the indexing process for each level of th, and report the mean time required to build each index. However, we do not follow this approach at this stage. To allow the reproducibility of the experiments, we report that Indri's indexing parameters `memory` was set to `50G` and `storeDocs` to `false`, across all indexing processes. The `memory` parameter provides a soft upper bound on the memory consumption of the indexer process (the total usage would be up to three times the parameter value). The `storeDocs` parameter set to `false` indicates that the original documents were not archived within the folder containing the Indri index.

On the contrary, we repeated the retrieval process 10 times for each setting. When reporting retrieval time statistics, we consider the mean value obtained from 10 iterations of the retrieval process.

The formats of the relevance assessments (i.e. qrels) for TREC 2009 and TREC 2010 ad-hoc tasks are different, as TREC 2009 qrels are suited for computing statMAP, while TREC 2010 qrels are tailored to standard MAP [6, 7]. This does not allow us to use a common set of ad-hoc measures across both topic sets. To overcome this issue and assess the retrieval effectiveness of the systems, we consider the TREC Web diversity task, although our systems do not perform any diversification of the document rankings. The diversity

[5]The actual percentages of pages that are considered as spam for $th = 20, 45, 70, 95$ are approximately $8\%, 22\%, 42\%, 80\%$, respectively.

[6]Note that $th = 70$ has been suggest to be the most effective threshold value [8].

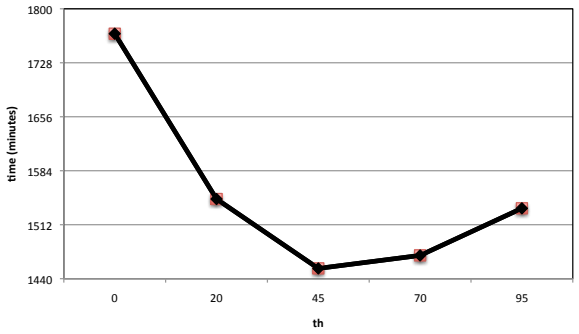

Figure 1: Indexing time for ClueWeb obtained using Systems S_1 and S_2 (which both correspond to value $th = 0$) and System S_3 for different values of the threshold th.

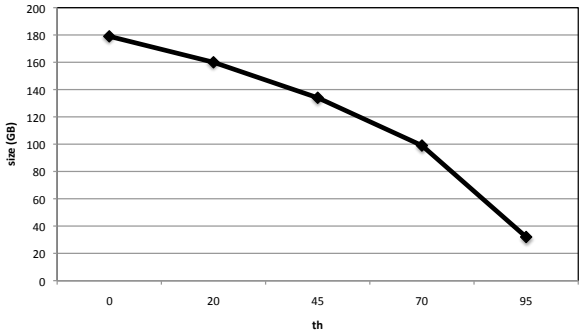

Figure 2: Size of the index of ClueWeb for Systems S_1 and S_2 (which both correspond to the value $th = 0$) and for System S_3 for different values of the threshold th.

qrels consist of the indication of document's relevance to a set of query-intents of a query, rather than to the query itself. The use of this retrieval task enables us to consider a common set of measures on the TREC 2009 and 2010 topic sets; in particular, we report the retrieval effectiveness of systems combining the effectiveness measured on both topic sets. The measures employed to assess the retrieval performance of systems are ERR-IA@10 [4] and α-nDCG@10 (with $\alpha = 0.5$) [5]. To provide an indication of how systems would perform in the task of standard ad-hoc retrieval, we produced standard qrels suited for computing (standard) MAP by considering a document relevant to a query if it is relevant to one or more subtopics of that query. MAP is then reported along with the selected diversity measures.

5.2 Results and Discussion

5.2.1 RQ1: Indexing Process

In figure 1 we report the time required for indexing the ClueWeb collection by Systems S_1, S_2 and S_3. For the latter system, we report the indexing time with respect to different settings of the threshold parameter th.

In answer to RQ1, removing spam during the indexing stage provides gains in terms of index efficiency, i.e. by lowering the indexing time, as demonstrated by the time required for indexing when using System S_3. In fact, System S_3 had been up to 20% times faster than System S_1 and System S_2 when indexing the ClueWeb collection. This result is not surprising, because System S_3's indexing process skips the documents that have been flagged as spam, and therefore System S_3 ends up processing less documents than Systems S_1 and S_2. However, note when using System S_3, the indexing time does not linearly decrease by increasing the value of the threshold th. Instead, high values of th for System S_3 may require more time for indexing the collection than that required by the same system for lower th. This is the case when $th = 95$: under this setting, S_3 requires 62 minutes more for indexing the collection than the same system with $th = 45$. This may be caused by the procedure we used for loading the file containing the list of documents which do not have to be considered for indexing. In fact, when $th = 95$ the list of documents to be excluded from indexing is large (998 MB against 271 MB for the list associated with $th = 45$) and the loading procedure may require more time than that saved for not indexing those documents.

5.2.2 RQ2: Indexes

In figure 2 we report on the size of the ClueWeb indexes as produced by Systems S_1, S_2 and S_3. For the latter system, we report the index sizes with respect to different settings of the threshold parameter th. Similarly, in table 1 we report the following index statistics: total number of indexed documents, total number of indexed terms, number of unique terms indexed.

In answer to RQ2, we found that the removal of spam during the indexing phase reduces the size of the index. Indexes created for the ClueWeb collection by System S_3 are up to 82.1% smaller than those created by Systems S_1 and S_2, when $th = 95$. As for question RQ1, this result is not surprising because in System S_3 up to 95% of the ClueWeb documents do not get indexed when $th = 95$: their statistics do not get recorded within the created index, and therefore the dimension of the index is smaller when compared to the standard index obtain by Systems S_1 and S_2. Furthermore, note that the decrease in index size in not linear with the increase of the value of threshold th: higher values of th provide higher index resizing ratios than lower ones. This is consistent with the statistics observed in table 1.

5.2.3 RQ3: Retrieval

The observations in sections 5.2.1 and 5.2.2 lead to the claims that filtering spam at indexing time provides advantages in terms of lowered indexing time (for some settings of th) and decreasing indexing size (consistently over all settings of th). But:

- does this translate into a reduced retrieval time as well?
- and, do these benefits harm retrieval effectiveness? or conversely, also is retrieval effectiveness

Systems	Statistics		
	Tot. # Indexed Documents	Tot. # Indexed Terms	Tot. # Indexed Unique Terms
S_1, S_2	50,220,423	40,417,956,329	87,331,162
S_3, $th = 20$	46,432,700	36,655,900,328	61,903,774
S_3, $th = 45$	39,303,448	30,833,198,931	48,382,773
S_3, $th = 70$	29,038,220	22,814,917,151	33,315,189
S_3, $th = 95$	10,008,217	7,278,840,138	11,532,356

Table 1: Index statistics (total number of indexed documents, total number of indexed terms, number of unique terms indexed) for the indexes of ClueWeb created by Systems S_1, S_2 and S_3.

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.0878	.1570	.1771	.1922	.1591	.1569	.1772	.1926	.1564
α-nDCG@10	.1495	.2427	.2571	.2666	.2195	.2425	.2571	.2669	2168
MAP	.1693	.1891	.1792	.1376	.0491	.1901	.1821	.1411	.0505

Table 2: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using LMDIR.

enhanced by considering a system that removes spam at indexing time?

These issues were explored when answering RQ3.

In figures 3 and 4 we plot the retrieval time of the four considered retrieval methods when using S_1 (specific case of S_3 with $th = 0$), S_2 and S_3, for varying values of the threshold th. For LMDIR, LM2S and BM25, there are marginal differences in retrieval time between systems S_1 and S_3, with respect to all values of th. On the contrary, differences are found for LMJM, where indexes created with higher th support faster retrieval, with $th = 95$ achieving a 93% speed-up with respect to system S_1 (8 seconds against 46 seconds). Retrieval times required by system S_2 are far greater than those required by S_1 and S_3. This is because in S_2 the removal of documents containing spam is performed at retrieval time. This process requires to perform a first pass of retrieval according to S_1. Then, the list of documents to be removed is loaded in memory, and documents are compared against the list. Documents considered as spam are then removed to only retrieve documents that are characterised by spam-scores greater than th. As the figures show, at parity of th, this process is considerably slower than using S_3: for example, for $th = 0$, retrieving using S_2 and LMJM is almost 364 times slower than the correspondent S_3 setting.

In table 2 we report the retrieval performance in terms of ERR-IA@10, α-nDCG@10 and MAP obtained by the three systems with different values of th and when using LMDIR. Similarly, tables 3, 4 and 5 report the retrieval performance obtained when using LMJM, LM2S and BM25, respectively.

The reported results show that retrieval effectiveness as measured by ERR-IA@10 and α-nDCG@10 is maximum when considering Systems S_2 and S_3 with

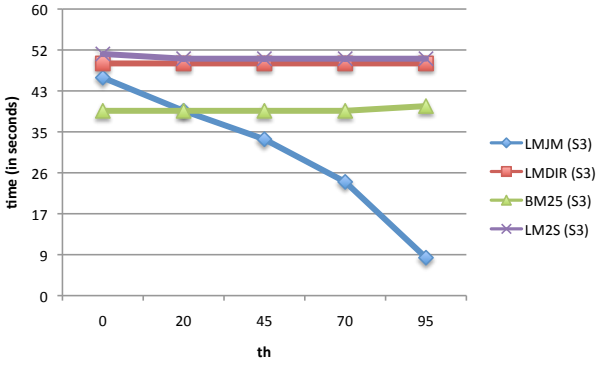

Figure 3: Retrieval time for 100 TREC 2009 and 2010 Web Track topics obtained using Systems S_1 (corresponding to the value $th = 0$) and S_3 for different values of the threshold th and for different standard IR retrieval models.

$th = 70$, regardless of the retrieval model implemented (i.e. LMDIR, LMJM, LM2S, BM25), while usually System S_1 obtains the lowest performance. In particular, we found that removing too many documents because of stringent spam threshold (i.e. $th = 95$), either during indexing or at post-retrieval phase, provides worse retrieval effectiveness than removing documents with $th < 70$. However, when a value of the threshold greater than 70 is used, Systems S_2 and S_3 yet generally obtained better retrieval performance than System S_1. When MAP is considered, best performance are obtained by low values of th (e.g. $th = 20$). However, for LMJM and BM25, no improvements in MAP are found for any setting of th.

When compared across the same levels of spam threshold th, Systems S_2 and S_3 exhibit similar retrieval performance. Specifically, S_3 is slightly more effective than S_2 when $th = 70$ is used, while S_2 is more effective than S_3 when $th = 95$. However,

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.0804	.0846	.0913	.1105	.0815	.0846	.0913	.1105	.0807
α-nDCG@10	.1262	.1368	.1411	.1543	.1178	.1368	.1411	.1543	.1166
MAP	.0905	.0866	.0775	.0589	.0237	.0870	.0791	.0614	.0255

Table 3: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using LMJM.

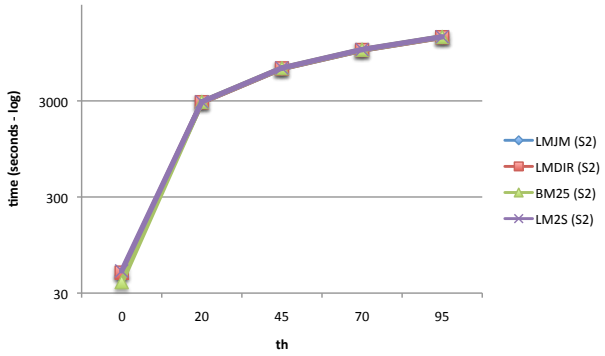

Figure 4: Retrieval time for 100 TREC 2009 and 2010 Web Track topics obtained using Systems S_1 (corresponding to the value $th = 0$) and S_2 for different values of the threshold th and for different standard IR retrieval models. Time in seconds is reported in logarithmic scale. Note that the retrieval times for S_2 are dominated by the process of loading in memory the file containing the list of spam documents.

differences are small and not statistically significant, when analysed using a two-tailed paired t-test.

In answer to RQ3, we found that indexing without spam does not harm retrieval effectiveness. Instead, the retrieval effectiveness of System S_3, which implements the indexing without spam procedure, is higher than that of System S_1 for specific values of th. We also found that System S_3 delivers similar retrieval performance than System S_2 when considering the same spam threshold th. However, when the retrieval times are considered, we found that S_1 is significantly faster than S_2. System S_3 generates indexes that are smaller than those of S_1: in general, this does not influence retrieval time for LMDIR, LM2S, and BM25. However, the retrieval time required when using LMJM decreasing with the increase of th. This difference in retrieval times may may be due to the specific implementation of the considered retrieval models in Indri. We also found that the time required for creating indexes when using System S_3 was generally lower than that required for Systems S_1 and S_2.

6 Conclusions

In this paper we have proposed a modification of traditional indexing procedures, called indexing without spam, where only documents considered as not containing spam are indexed. We have shown that this approach modifies the index statistics when compared to traditional indexing procedures, and we have linked our approach with methods of index pruning, and in particular with [20].

To assess the effectiveness of our proposal, we set up a thorough empirical comparison using the TREC ClueWeb collection and the TREC 2009 and 2010 Web topics. Results suggest that indexing without spam is more effective than standard indexing procedures combined with post-retrieval spam removal, when considering retrieval measures such as ERR-IA, α-nDCG and MAP.

An important observation is that indexing pruning strategies are believed to harm retrieval performance, e.g. [20], while obtaining gains in throughput. However, empirical results demonstrate that indexing without spam prunes indexes without harming retrieval effectiveness: often our approach delivers better performance than the other considered strategies.

Future work aims to extend this study by considering:

- more executions of the indexing procedures, so as to obtain more reliable indications of the time required for indexing in each experimental setting;
- ClueWeb part A, which consists of more than one billion web pages. In particular, it is interesting to explore whether the index statistics obtained through the indexing without spam procedure are sensibly more effective then those collected by standard indexing procedures when used for retrieving documents.
- a more systematic exploration of how document removal affects retrieval effectiveness.

References

[1] Roi Blanco and Álvaro Barreiro. Static pruning of terms in inverted files. In *ECIR '07*, pages 64–75. 2007.

[2] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek and Aya Soffer. Static index pruning for information retrieval systems. In *SIGIR '01*, pages 43–50, 2001.

[3] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock and Fabrizio Silvestri. Know your neighbors: web spam detection using the

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.0878	.1570	.1771	.1922	.1591	.1569	.1772	.1926	.1564
α-nDCG@10	.1495	.2427	.2571	.2666	.2195	.2425	.2571	.2669	.2168
MAP	.1693	.1891	.1792	.1376	.0491	.1901	.1821	.1411	.0505

Table 4: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using LM2S.

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.1452	.1824	.2016	2026	1531	1831	.2015	2035	.1496
α-nDCG@10	.2207	.2636	.2720	.2740	.2098	.2653	.2715	.2756	.2056
MAP	.1528	.1492	.1338	.0984	.0390	.1498	.1360	.1012	.0392

Table 5: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using BM25.

web topology. In *SIGIR '07*, pages 423–430, 2007.

[4] Olivier Chapelle, Donald Metlzer, Ya Zhang and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09*, pages 621–630, 2009.

[5] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Buttcher and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*, pages 659–666, Singapore, 2008.

[6] C.L. Clarke, N. Craswell and I. Soboroff. Overview of the trec 2009 web track. In *Proc. of TREC 2009*, 2009.

[7] C.L.A. Clarke, N. Craswell, I. Soboroff and G.V. Cormack. Overview of the trec 2010 web track. In *Proc. of TREC 2010*, 2010.

[8] Gordon Cormack, Mark Smucker and Charles Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *JIR*, pages 1–25, 2011.

[9] Claudia Hauff and Djoerd Hiemstra. University of twente @ trec 2009: Indexing half a billion web pages. In *Proc. of TREC 2009*, 2009.

[10] J. He, K. Balog, K. Hofmann, E. Meij, M. de Rijke, M. Tsagkias and W. Weerkamp. Heuristic ranking and diversification of web documents. In *Proc. of TREC 2009*, 2009.

[11] Rianne Kaptein, Marijn Koolen and Jaap Kamps. Result diversity and entity ranking experiments: Anchors, links, text and wikipedia. In *Proc. of TREC 2009*, 2009.

[12] J. Lin, D. Metzler, T. Elsayed and L. Wang. Of Ivory and Smurfs: Loxodontan MapReduce Experiments for Web Search. In *Proc. of TREC 2009*, 2009.

[13] Alistair Moffat and Justin Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.*, Volume 14, pages 349–379, October 1996.

[14] Alexandros Ntoulas, Marc Najork, Mark Manasse and Dennis Fetterly. Detecting spam web pages through content analysis. In *WWW '06*, pages 83–92, 2006.

[15] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. and Tr. in IR*, Volume 3, Number 4, pages 333–389, 2009.

[16] K. Sparck-Jones, S. Walker and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *IP&M*, Volume 36, Number 6, pages 779 – 808, 2000.

[17] T. Strohman, D. Metzler, H. Turtle and W. B. Croft. Indri: A language model-based search engine for complex queries. *ICIA '04*, 2004.

[18] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342, 2001.

[19] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *SIGIR '02*, pages 49–56, 2002.

[20] Lei Zheng and Ingemar Cox. Entropy-based static index pruning. In *ECIR '09*, pages 713–718. 2009.