Improving Real Time Search Performance using Inverted Index Entries Invalidation Strategies

Esteban A. Ríssola and Gabriel H. Tolosa

Departamento de Ciencias Básicas, Universidad Nacional de Luján, Luján, Buenos Aires, Argentina

{earissola, tolosoft}@unlu.edu.ar

Abstract

The impressive rise of user-generated content on the web in the hands of sites like Twitter imposes new challenges to search systems. The concept of real-time search emerges, increasing the role that efficient indexing and retrieval algorithms play in this scenario. Thousands of new updates need to be processed in the very moment they are generated and users expect content to be "searchable" within seconds. This lead to the develop of efficient data structures and algorithms that may face this challenge efficiently.

In this work, we introduce the concept of *index entry invalidator*, a strategy responsible for keeping track of the evolution of the underlying vocabulary and selectively *invalidate* and *evict* those inverted index entries that do not considerably degrade retrieval effectiveness. Consequently, the index becomes smaller and may increase overall efficiency. We introduce and evaluate two approaches based on Time-to-Live and Sliding Windows criteria. We also study the dynamics of the vocabulary using a real dataset while the evaluation is carry out using a search engine specifically designed for real-time indexing and search.

Keywords: real time search, index entry invalidator, efficiency.

1 Introduction

The impressive rise of social media and other forms of user-generated content during last decade in the hands of sites like Twitter or Facebook [1, 2, 3, 4, 5, 6] reveals us the compelling challenge that traditional search must face. This growth is not only defined by the number of users who consumes these services but also by the vast amount of content they produce (*i.e.*, eight tweets/second on average [6]). The implications that the concept of *real-time* introduces give us a hint about the significant role that efficient indexing and retrieval algorithms plays in this scenario.

Search and retrieval over this huge collections, as well as the management of the involved data structures exhibit some differences and introduce new requirements in comparison to classical Web search operations [7, 8]. On the one hand, both queries and user behaviour differ from traditional patterns [5, 9]. On the other hand, real-time search service becomes very challenging in large-scale microblogging systems where thousands of new updates need to be processed in the very moment they are generated. Indexing can not be considered as a batch operation any more as users expect content to be available (*searchable*) within seconds. Thereby, the indexer should be designed to receive a continuous stream of documents (at very high arrival rates, often with sudden spikes) and to achieve both low latency and high throughput. Finally, the nature of real-time search means that temporal stamps are important for ranking, beyond the application of other metrics aimed to improve the quality of the result list.

Our particular focus is over microblog services, like Twitter, where users are able to write brief status messages called *posts*[1] that can share with their network of friends and, often, with the general public at the very moment they are generated. Increasingly, this kind of services grows in popularity and therefore, the data volume they have to deal with becomes larger every day. As far as we know, the only practical strategy to cope with the performance requirements cited above consists in maintaining the inverted index and its corresponding structures in main memory [7, 8]. This strategy primarily admit to significantly reduce reading and writing latencies as compared to other devices, such as hard disks. Nevertheless, memory remains today a scare resource [7] such that becomes essential to ascertain the way to store the index only the necessary information to provide reasonable (or acceptable) effectiveness.

Thus, bearing in mind the context and its inherent requirements we propose the development of a component called *index entries invalidator*, responsible for keeping track of the evolution that the underlying vocabulary frequency patterns. It aims to selectively *invalidate* and *evict* those inverted index's entries whose absence won't considerably degrade retrieval effectiveness. Consequently, the index becomes smaller and may increase overall efficiency. To the best of our knowledge, this is the first work that faces this problem and proposes

[1] Throughout this work the terms post, document or tweet are used interchangeable.

the concept of index entries invalidator for real time systems.

In order to design an efficient invalidation algorithm we conduct the analysis of a real sample of Twitter data to understand its growth dynamic. Specifically, we employ the Tweets2011 [10] dataset widely used by scientific community in this field. Moreover, we select a sample of queries from the well-known AOL Query Log [11]. Our experiments examine the performance of the retrieval process and the effect on the quality of the results. In this paper we extend a previous work [12] where only a time-to-live based invalidation strategy is introduced.

Our contributions are as follows: (a) We look into the vocabulary obtained from the tweets dataset and study its underlying dynamics. Furthermore, we identify tree types of tokens and show that the size of the resulting vocabulary can not be fitted by the Heaps' law [13], as traditional collections; (b) We design and build up *index entries invalidators* inspired by the concepts of cache invalidators [14], based on the time that the entries have persisted in the index without been updated (besides the time-to-live strategy introduced in [12], we design a new invalidator based on a sliding window approach); (c) We perform the corresponding evaluations making use of a self-modified version of Zambezi [15] in-memory search engine for streaming documents by implementing the proposed invalidators. We measure both wall-clock time and effectiveness metrics on a per-query basis.

The remainder of the work is organized as follows: Section 2 provides background concepts on real-time index features. Section 3 describes the employed collection and introduces vocabulary dynamics analysis. Section 4 presents the index entries invalidator approach. Section 5 details experiments and results. Section 6 concludes and introduces future work.

2 Background and Related Work

Information retrieval systems rely on efficient data structures to support search, the so-called inverted index [16]. Basically, it stores the set of all unique terms in the document collection (*vocabulary*) associated to a set of entries that form a posting list. Each entry represents the occurrence of a term t within a document d and it consists of a document identifier (DocID) and a payload that is used to store information about the occurrence of t within d. Each posting list is sorted in an order that depends on the specific query resolution strategy [17, 18, 16]. One of the key features of real-time search resides in the fact that new contents should be available for search immediately after their creation, while concurrently supporting low-latency, high throughput query evaluation. This implies that the index needs to be update incrementally as new documents arrive to the system. For this reason, the indexing process requires allocating space for postings in a dynamic fashion, that results in non-contiguous postings lists [19].

Nowadays, Twitter's Earlybird retrieval engine [7], built upon this large scale microblogging service specific needs, represents a point in the space of real-time search engines. According to its design guidelines, a general solution to the problem of dynamically allocating postings for real-time search is proposed in [8]. As Earlybird represents a particular instantiation, they provide a general framework for incremental indexing where all data structures are stored completely in memory. Thus, from a small number of memory pools, increasingly larger slices for postings are allocate as more term occurrences are encountered. This solution is planned not only for indexing tweets but also it is aligned to applications that have really tight index latency requirements.

Likewise, several approaches have been proposed in the literature to improve indexing and ranking phases, in terms of efficiency and effectiveness. Chen et al. [3] introduced an adaptive indexing scheme aimed at reducing the update cost by delaying indexing less useful tweets (*i.e.*, tweets that may not appear in the search results). Otherwise, they are grouped with other unimportant tweets and indexed later as part of an offline batch process. In [20] an online topic modeling framework for querying large microblog corpus is presented. Such models were employed to identify topics in the tweets and compare them with the ones obtained from the incoming queries. Furthermore, discovered topics are applied to rank relevant tweets in the collection. This approach is called *online* in the sense that corresponding topic modeling was not only conducted over hourly batches of captured tweets in an offline fashion, but also for recent time intervals that has not yet been included in the last batch.

Moreover, [21, 22] have also proposed strategies to improve the overall effectiveness of microblog retrieval systems. In the former, Choi & Croft suggested to extend a previously defined time-based model for pseudo-relevance feedback query expansion, by incorporating the temporal factor into ranking. In particular, they claim that selecting relevant time period for a specific query based on a user behaviour that can be collected easily, like retweeting, and extracting expanded terms by using weights derived from the relevant time can improve retrieval performance. On the other hand, Metzler et al. define a search task called event retrieval, *i.e.*, given a query that describes

a particular event the intention is to retrieve a ranked list of structured event representations. These correspond to a series of timespans during which an instance of the event occurred. An unsupervised methodology is proposed to extract high quality event representations by applying a temporal query expansion technique.

Most recently, Nepomnyachiy et al. [6] introduce a search framework for geo-temporally tagged data to support low-latency retrieval for queries with spatial, temporal, and textual components. Mainly, they define an efficient way to organize index content based on the spatial distribution of user-generated data and considering documents timestamp.

These works tackle different ways to organize index structures in order to boost retrieval performance for real-time search systems. However, they do not consider the possibility of invalidate entries based on the idea of terms' update frequency reducing the index resulting size, thus allowing a more efficient utilization of available computing resources.

3 Vocabulary Dynamics

In this Section we introduce an specific analysis of the vocabulary dynamics using the Tweets2011 dataset, that is employed in different works [21, 8, 15] related to real time search.

Data Characterization: Tweets2011 [10] is constituted as the reference collection of the TREC-2012 Microblog Track [23]. It comprises of approximately 16 million tweets crawled between January 23^{rd} and February 8^{th}, 2011. This dataset is designed to be a reusable, representative sample of the twittersphere. We successfully downloaded 11,601,066 tweets, filtering out all non-English posts. The tweets distribution over time is shown in Figure 1, on average $9,3$ documents arrive per hour. Each post is composed of roughly 13.39 useful words and 81.25 characters. As stated in [6], the number of words in a tweet is considerably small and tokens rarely repeat within a document.

Dynamics: Taking into account the context of the proposed analysis, we decided to split each post considering three types of tokens, namely: mentions("@"), hashtags("#"), and general terms. The reason for applying this convention lies behind the fact that both mentions and hashtags have a particular meaning and value inside Twitter [4, 5]. According to traditional IR literature, a practical way to describe how vocabulary and collection size are related corresponds to Heaps' law [13] expressed as $v = k \times n^{\beta}$, where k and β represent constant values related to the input text while n is the total number of processed tokens. The output, v, becomes the total number

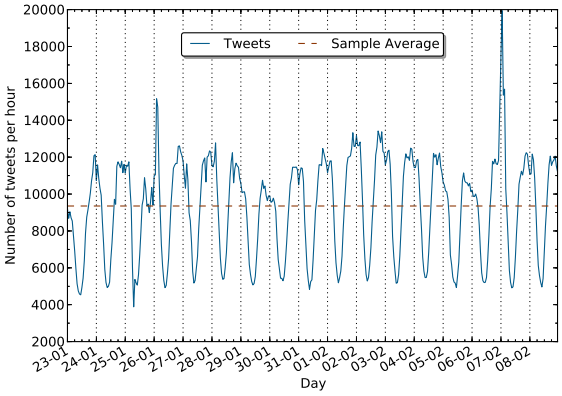


Figure 1: Tweets arrival rate

of unique tokens in the collection. In particular, it enables to empirically estimate vocabulary size (and its growth) as a function of the collection size. Figure 2 shows that the considered tokens growth faster than this law predictions, rather exhibiting a linear growth (similarly, the whole vocabulary). Despite the number of new encountered tokens raises considerably fast (daily, about a 72% of the vocabulary corresponds to new additions), mostly of those tokens are *hapax*, *i.e.*, they appear only once within a context. They represent roughly 67.7% of the whole vocabulary.

As stated in [21, 22, 6], this phenomenon is due to the informal essence that distinguish microblogging activity along with its character limit (140 in Twitter's case). Therefore, abbreviations, elongated words, compound words hashtags, internet slangs and misspellings are common, in many cases deliberate.

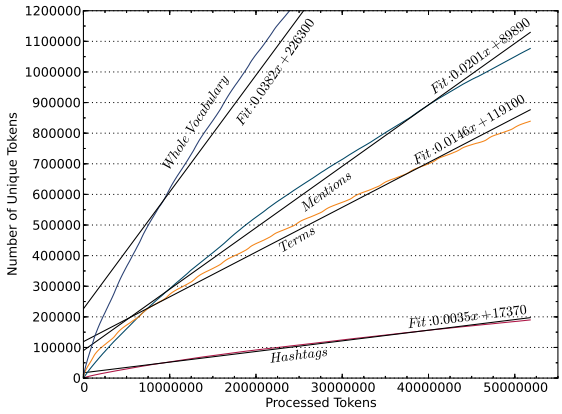


Figure 2: Vocabulary growth

In order to proceed with vocabulary dynamics characterization, we classify tokens in three groups by studying how their frequency evolves across the days. To this extent, we apply a *sliding window* approach of s slots. Each slot corresponds to the observation of a token on a daily basis. We set a boolean true value if a token is present in at least one document of the corresponding day. Ev-

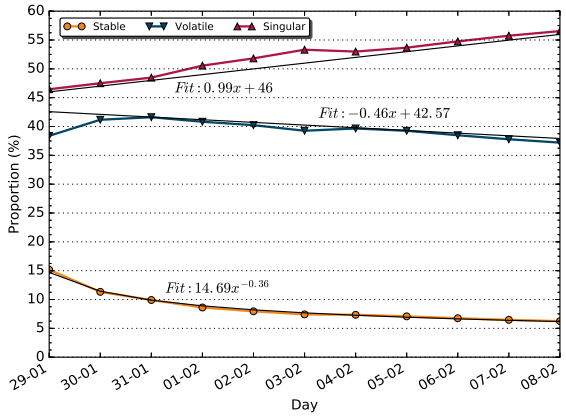


Figure 3: Proportion of each type of token over days with the corresponding fit model

ery day, tokens' frequency information is updated according to the ingested tweets. When a token appears for the first time (*i.e.*, it does not exist in the current vocabulary) a new instance of the window is assigned to it and the first slot activates. Thus, as days move forward the window goes over the remaining slots and these may activate (or not) following token's daily frequency behaviour. When the window walks through s slots, a side shift is applied maintaining the last observations. Then, we classify each token according to the following criteria:

Let w_i be the window of s slots that corresponds to token t_i, and w_{ij} be its value at slot j. Let $S_i = \sum_{j=1}^{s} w_{ij}$ and $G(t_i)$ a function that assigns a category (or group) to each token according its occurrence behaviour, defined as:

$$G(t_i) = \begin{cases} stable, & \text{if } S_i = s \\ volatile, & \text{if } 2 \leq S_i \geq (s-1) \\ singular, & \text{if } S_i = 1 \end{cases}$$

In our study we apply a window $s = 7$ (a whole week) that we consider as a reasonable number to study how tokens behave. Figure 3 shows the distribution of the classified tokens across the sample. Note that we are able to start with this labeling on January 28^{th}, because until then the window has not reached the 7^{th} slot.

Tokens categorized as "singular" grow practically in a linear fashion, as they present a slope of 0.99. In addition, they exhibit a growth rate of roughly 1%. "Volatile" tokens also show a linear behaviour (dismissing the first observation), in this case with a slope of -0.46 and a daily growth close to 0.98%. Finally, "stable" tokens fit to a power law with $\beta = 0.36$, as they decrease at a daily rate of about 10%. In particular, this last group shows that despite new tokens are continuously appended to the vocabulary, a considerably smaller subset of them prevails over days.

4 Index Entries Invalidators

According to the preceding analysis it is reasonable to think about pruning some entries of the inverted index. Specifically, those tokens whose daily frequency behaviour give us a hint about its scarce contribution to search results. Similarly to Lin & Mishne [24] observation, there is a great deal of "churn" in tweets content. To tackle this issue we decide to prune the inverted index by removing the full posting lists of those tokens that sparsely appear over days. Therefore, we define an *index entries invalidator* (IEI) by applying two approaches:

TTL-based: The first one corresponds to a time-to-live (TTL) strategy, as adopted in result caches [25]. This IEI is based on the time that entries have persisted in the index without been updated. When a token exceeds a given threshold, the IEI invalidates and evicts this one along with its posting lists. In other words, an entry is dismissed when the difference between the current time and the last update time is larger than the TTL value. Note that lower values might diminish effectiveness, as the frequency in which tokens occur in subsequent document should be enough to avoid its TTL to expire. Hence, the chosen TTL threshold should offer a tradeoff between index size (efficiency) and effectiveness. This strategy guarantee that all tokens *live* in the index for at least n hours (the TTL value) in contrast to adopt a frequency criterion pruning at certain time in the day.

SW-based: The other one is based on the sliding window (SW) methodology described in Section 3. Tokens categorized as "singular" are eliminated from the index, while a tolerance threshold between 2 y $(s-1)$ is defined for "volatile" ones. That is, tokens whose $G(t_i) \leq THRS$ are also pruned from the vocabulary. To some extent, this setting enables to determine the pruning aggressiveness, as values closer to s suggest the elimination of more entries. Recall that the classification process is able to start once the window has walked through s slots, triggering only if the token has not appeared in the last day.

5 Experiments and Results

Methodology: In order to evaluate these approaches in a real time search scenario, we modify Zambezi search engine, whose source code is publicly available. We implement the index entries invalidators to perform different experiments to determine the efficiency and effectiveness of the proposed strategies. For the TTL-based invalidator we set the value to 24 hours in order to check whether an index entry has expired. By taking advantage of tweets timestamps, we are able to

establish the beginning and end of the days, in terms of hours. Thus, every 24 hours during the 2 weeks covered by the sample we browsed the index and evict the corresponding entries according to the heuristic previously defined. After every day, we processed 1 million queries measuring efficiency and effectiveness by comparing the results obtained from the original index without pruning (our baseline), with those of the pruned one. We run three series of experiments retrieving the top-k documents that are most relevant to a query (with $k = \{50, 10, 100\}$). We configure Zambezi to use its WAND [26] algorithm adaptation for microblogging. Essentially, it uses timestamps as sorting criteria along with a simplified version of the scoring BM25 function. Finally, to ensure the consistency of the results, we perform five trials of the experiment, and average the outcomes. The same methodology is applied to evaluate the performance of the sliding window strategy, employing three different threshold values ($\{2, 4, 6\}$). Due to lack of a publicly available real-time query set various works build up synthetic ones. Initially, we evaluate our approaches with a shred of the synthetic query log generated in [6]. However, the number of queries in the set is not enough to run a robust performance evaluation. For this reason, we decide to use one million queries extracted from the well-known AOL Query Log [11], employed in other works involving real-time search [24, 8].

Metrics: To assess the overall performance of the approaches we evaluate both efficiency (time and space) and effectiveness. In the first case, the execution time is measured in terms of wall-clock time on a per query basis. We also evaluate the number of invalidated entries per day and the index size reduction. To quantify the effectiveness, we apply the result set intersection between the baseline and our approaches. Remember that in a real-time search scenario one of the primary search task consist in presenting the most recent documents related to the query (recency or freshness of the results) [27].

Results: The effectiveness evaluation shows that the TTL-based IEI does not degrade the results significantly. Figure 4 exhibits the intersection ratio (averaged from 1M queries) for the three series of experiments. In the case of top-10 retrieval, less than one document (on average) is missing in the pruned result set. This result is proportionally similar in the remaining series (top-50 and 100). A deep analysis of pruned tokens explains that most of them correspond to rare ones that appear sparsely in queries.

Regarding efficiency, this approach reduces the overall execution time in all configurations up to 6% (best case). Figure 5 shows the results for top-10 posts (we omit $k = \{50, 100\}$ figures due to lack of space but results perform similarly). The increased execution time over days corresponds to the vocabulary growth. The difference also increases while tokens are added to the index, thus more tokens are invalidated and pruned.

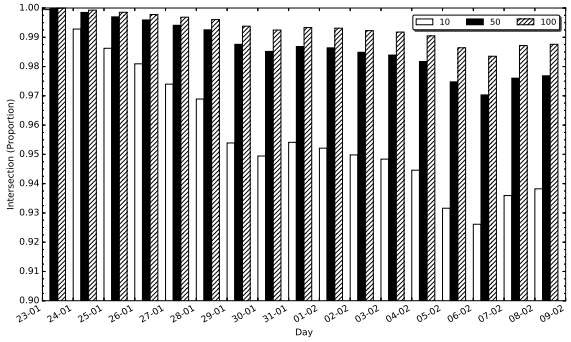


Figure 4: Intersection proportion when retrieving top 10, 50 and 100 documents.

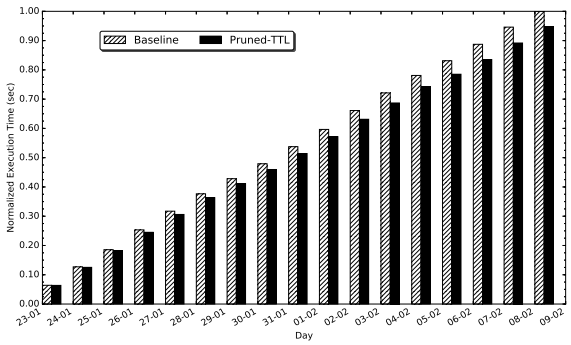


Figure 5: Normalized Execution Time for TTL-based Invalidator (Top-10 configuration)

The space consumed by the inverted index in both baseline and pruned versions is also analyzed. Table 1 shows the number of entries in the vocabulary. The number of valid entries decreases dramatically over days (up to 88%). This enables faster lookups into the vocabulary. However, the total number of DocIDs that accounts for the resulting posting lists decreases much slower (Table 2), up to 9.1% in the last day. Again, this happens due to the pruned tokens correspond to infrequent ones whose document frequency is very low (typically, one). This result suggests that a more aggressive pruning strategy that considers the posting list lengths of the tokens that remain into the vocabulary may lead to greater benefits.

Concerning the sliding window approach we observe that the effectiveness is not practically hurt. The results for SW-THRS=2 perform similarly than the baseline (we omit the details) so a larger threshold is required. Figure 6 depicts the intersection ratio for the different values of k and thresholds $\{4, 6\}$. For $k = 10$ one document (on average) is missing from the retrieved set. Simi-

Day	Baseline	Pruned-TTL	Diff. %
1	217,044	209,379	3.53
2	390,863	231,976	40.65
3	540,456	238,330	55.90
4	692,677	259,240	62.57
5	833,499	250,156	69.99
6	970,939	256,852	73.55
7	1,081,271	223,664	79.31
8	1,186,153	228,312	80.75
9	1,295,470	239,079	81.54
10	1,407,580	258,560	81.63
11	1,519,236	254,509	83.25
12	1,627,904	258,206	84.14
13	1,736,197	259,738	85.04
14	1,827,029	227,970	87.52
15	1,916,499	230,474	87.97
16	2,012,312	245,609	87.79
17	2,105,807	249,280	88.16

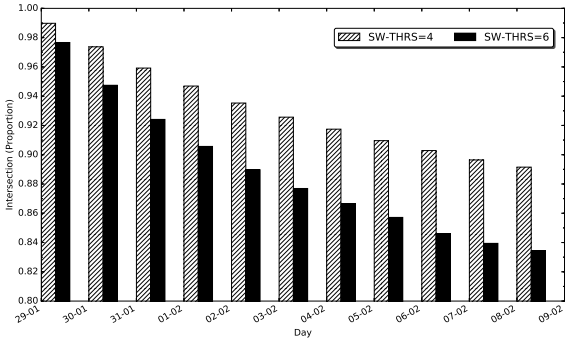
Table 1: Number of entries in the inverted index for TTL-based invalidator

Day	Baseline	Pruned-TTL	Diff. %
1	2,392,099	2,384,262	0.33
2	5,174,584	4,992,039	3.53
3	7,864,478	7,467,306	5.05
4	10,919,587	10,294,900	5.72
5	13,892,852	12,991,206	6.49
6	16,859,239	15,678,477	7.00
7	19,413,681	17,914,817	7.72
8	22,037,617	20,270,901	8.02
9	24,934,503	22,903,479	8.15
10	27,976,632	25,684,408	8.19
11	31,168,827	28,576,005	8.32
12	34,215,991	31,326,117	8.45
13	37,198,345	33,998,347	8.60
14	39,793,764	36,240,415	8.93
15	42,532,978	38,669,507	9.08
16	45,659,362	41,508,202	9.09
17	48,608,362	44,153,772	9.16

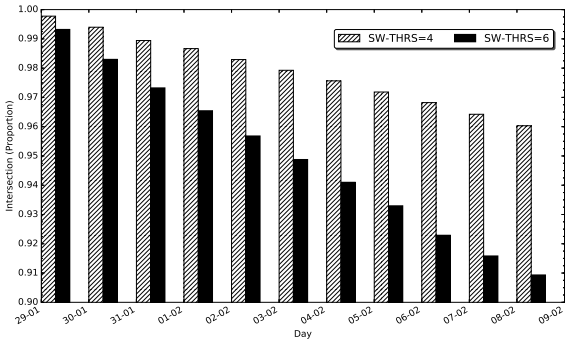
Table 2: Sum of DocIDs in the posting lists (all terms) for TTL-based invalidator

larly, the remaining configurations ($k = \{50, 100\}$) are not substantially affected by the invalidation process, highlighting that a more strict threshold does not degrade significantly the final ranking.

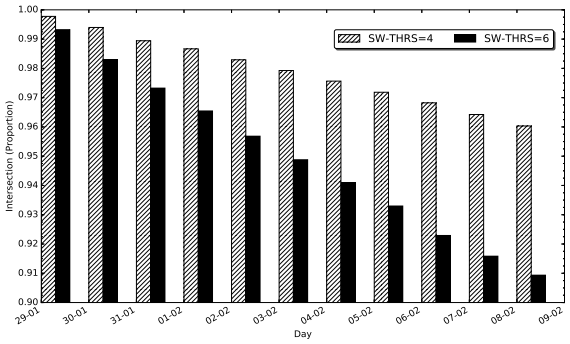
The efficiency evaluation reveals that this strategy improves the overall execution time in all configurations, reaching almost a 10 % for SW-THRS=6. For SW-THRS=4 the performance is close to the TTL approach (6,41%). Figure 7 shows these results (again, due to space constraints we omit $k = \{50, 100\}$ figures, nonetheless results perform similarly). Table 3 shows that about 69.7% and 70.5% of space could be saved by pruning the index entries by applying this methodology (THRS=$\{4, 6\}$, respectively). On the other hand, the total number of DocIDs can be reduced on about 7.26% and 8.7% (Table 4). As we aforementioned for TTL invalidator, a supplementary strategy considering posting lists pruning of the remaining tokens will be adequate to deal with the vocabulary growth.



(a) Top-10 results



(b) Top-50 results



(c) Top-100 results

Figure 6: Intersection proportion when retrieving top 10, 50 and 100 documents.

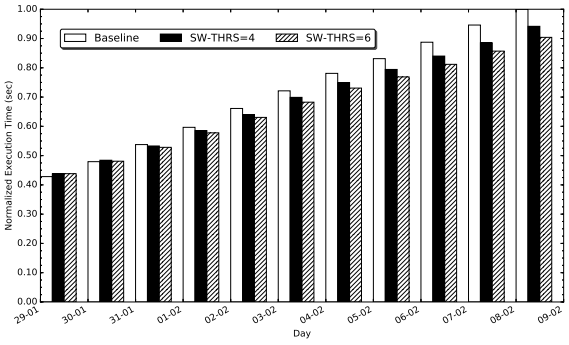


Figure 7: Normalized Execution Time for SW-based Invalidator (Top-10 configuration)

6 Conclusions and Future Work

In this work we introduced the concept of index entries invalidator, an strategy for real time search scenarios, that aims to selectively invalidate and

Day	Baseline	SW-THRS=4	SW-THRS=6
7	1,081,271	934,260 (-13.60%)	928,040 (-14.17%)
8	1,186,153	873,004 (-26.40%)	861,738 (-27.35%)
9	1,295,470	830,676 (-35.88%)	816,633 (-36.96%)
10	1,407,580	788,382 (-43.99%)	772,451 (-45.12%)
11	1,519,236	754,785 (-50.32%)	737,668 (-51.44%)
12	1,627,904	722,069 (-55.64%)	704,236 (-56.74%)
13	1,736,197	715,801 (-58.77%)	697,567 (-59.82%)
14	1,827,029	698,237 (-61.78%)	679,515 (-62.81%)
15	1,916,499	676,089 (-64.72%)	657,298 (-65.70%)
16	2,012,312	658,136 (-67.29%)	639,921 (-68.20%)
17	2,105,809	638,118 (-69.70%)	620,422 (-70.54%)

Table 3: Number of entries in the inverted index for SW-based invalidator

Day	Baseline	SW-THRS=4	SW-THRS=6
7	19,413,681	19,152,954 (-1.34%)	19,080,321 (-1.72%)
8	22,037,617	21,432,916 (-2.74%)	21,278,217 (-3.45%)
9	24,934,503	23,990,685 (-3.79%)	23,771,701 (-4.66%)
10	27,976,632	26,685,536 (-4.61%)	26,400,001 (-5.64%)
11	31,168,827	29,534,438 (-5.24%)	29,181,683 (-6.38%)
12	34,215,991	32,239,236 (-5.78%)	31,825,493 (-6.99%)
13	37,198,345	34,905,933 (-6.16%)	34,435,359 (-7.43%)
14	39,793,764	37,209,193 (-6.49%)	36,671,148 (-7.85%)
16	42,532,978	39,643,331 (-6.79%)	39,040,825 (-8.21%)
16	45,659,362	42,447,901 (-7.03%)	41,800,241 (-8.45%)
17	48,608,362	45,077,120 (-7.26%)	44,380,423 (-8.70%)

Table 4: Sum of DocIDs in the posting lists (all terms) for SW-based invalidator

evict those inverted index entries that do not considerably degrade retrieval effectiveness. Consequently, the index becomes smaller thus increasing the overall efficiency. Our experimental results showed that the proposed approaches reduce the number of entries in the vocabulary up to an 88% (TTL-based), enabling faster lookups. The overall execution time for our query-set is also reduced up to 10% (SW-based).

However, the resulting size of the index decreases much slower. In order to deal with this issue, we plan to extend the IEI by pruning at both entry and posting list levels. To this extent, it will be necessary to consider the posting list lengths of the tokens that remain into the vocabulary and to study how they evolve over time. Moreover, we are interested to expand the family of invalidators improving them by the application of different pruning techniques adapted to this problem.

Acknowledgements

We give special thanks to the CIDETIC (Centro de Investigación Docencia y Extensión en TIC, UNLu. `http://cidetic.unlu.edu.ar/`) for providing us the necessary computational resources in order to conduct the corresponding experiments.

References

[1] B. A. Huberman, D. M. Romero, and F. Wu, "Social networks that matter: Twitter under the microscope," *CoRR*, vol. abs/0812.1045, 2008.

[2] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, 2010.

[3] C. Chen, F. Li, B. C. Ooi, and S. Wu, "Ti: An efficient indexing mechanism for real-time search on tweets," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, 2011.

[4] M. Efron, "Information search and retrieval in microblogs," *J. Am. Soc. Inf. Sci. Technol.*, vol. 62, no. 6, 2011.

[5] J. Teevan, D. Ramage, and M. R. Morris, "#twittersearch: A comparison of microblog search and web search," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, 2011.

[6] S. Nepomnyachiy, B. Gelley, W. Jiang, and T. Minkus, "What, where, and when: Keyword search with spatio-temporal ranges," in *Proceedings of the 8th Workshop on Geographic Information Retrieval*, GIR '14, 2014.

[7] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin, "Earlybird: Real-time search at twitter," in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, 2012.

[8] N. Asadi, J. Lin, and M. Busch, "Dynamic memory allocation policies for postings in real-time twitter search," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, 2013.

[9] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: Understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, WebKDD/SNA-KDD '07, 2007.

[10] R. McCreadie, I. Soboroff, J. Lin, C. Macdonald, I. Ounis, and D. McCullough, "On building a reusable twitter corpus," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, 2012.

[11] G. Pass, A. Chowdhury, and C. Torgeson, "A picture of search," in *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale '06, 2006.

[12] E. Ríssola and G. Tolosa, "Inverted index entry invalidation strategy for real time search," in *Proceedings of the XXI Congreso Argentino de Ciencias de la Computación*, CACIC '15, 2015.

[13] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[14] R. Blanco, E. Bortnikov, F. Junqueira, R. Lempel, L. Telloli, and H. Zaragoza, "Caching search engine results over incremental indices," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, 2010.

[15] N. Asadi and J. Lin, "Fast candidate generation for real-time tweet search with bloom filter chains," *ACM Trans. Inf. Syst.*, vol. 31, no. 3, 2013.

[16] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Comput. Surv.*, vol. 38, no. 2, 2006.

[17] R. A. Baeza-Yates and B. A. Ribeiro-Neto, *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England, 2011.

[18] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes (2Nd Ed.): Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc., 1999.

[19] N. Asadi and J. Lin, "An exploration of postings list contiguity in main-memory incremental indexing," LSDS-IR '14, 2014.

[20] C. E. Grant, C. P. George, C. Jenneisch, and J. N. Wilson, "Online topic modeling for real-time twitter search," in *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*, 2011.

[21] J. Choi and W. B. Croft, "Temporal models for microblogs," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, 2012.

[22] D. Metzler, C. Cai, and E. Hovy, "Structured event retrieval over microblog archives," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, 2012.

[23] I. Soboroff, I. Ounis, C. Macdonald, and J. Lin, "Overview of the trec-2012 microblog track," in *In Proceedings of TREC 2012*, 2012.

[24] J. Lin and G. Mishne, "A study of "churn" in tweets and real-time search queries," in *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[25] B. B. Cambazoglu, F. P. Junqueira, V. Plachouras, S. Banachowski, B. Cui, S. Lim, and B. Bridge, "A refreshing perspective of search engine caching," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, 2010.

[26] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien, "Efficient query evaluation using a two-level retrieval process," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, 2003.

[27] D. McCullough, J. Lin, C. Macdonald, I. Ounis, and R. McCreadie, "Evaluating real-time search over tweets," in *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.