Inverted Index Entry Invalidation Strategy for Real Time Search

Esteban A. Ríssola and Gabriel H. Tolosa

Departamento de Ciencias Básicas
Universidad Nacional de Luján
{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 study the dynamics of the vocabulary using a real dataset and also provide an evaluation of the proposed strategy using a search engine specifically designed for real-time indexing and search.

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 [12, 14, 8, 10, 23, 20] 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[1]. 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 extensive collections, as well as the management of the involved data structures present certain differences and introduce new requirements in comparison to classical Web search operations [6, 3]. On the one hand, because both queries and user behaviour differ from traditional [23, 13]. On the other hand, because real-time search service results indeed to be 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

[1] According to [20], an average of eight tweets are issued per second.

(*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. In addition, as documents are ingested the content of the corresponding structures need to be updated to allow the index to serve incoming queries. This implies the existence of concurrent operations that should be handled carefully. 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*[2] 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 [6, 3]. This strategy primarily admit to significantly reduce reading and writing latencies as compared to other devices, such as disks. Nevertheless, memory remains today a scare resource [6] 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 presents. 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. 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 [17] dataset widely used by scientific community in this field, composed roughly of 16 million tweets crawled during 2011. Moreover, we select 1 million queries from the well-known AOL Query Log[3] [21]. The experiments examine the performance of the retrieval process and the effect on the quality of the results.

Our contributions are as follows: (a) We look into the vocabulary obtained from our 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 [16], as traditional collections; (b) We design and build up an *index entries invalidator* inspired by the concepts of cache invalidators [5], based on the time that the entries have persisted in the index without been updated (time-to-live approach [7]); (c) We perform the corresponding evaluations making use of a self-modified version of Zambezi[4] [1] in-memory search engine for streaming documents by implementing the proposed invalidator. We measure both wall-clock time and effectiveness metrics on a per-query basis.

[2] Throughout this work the terms post, document or tweet are used indistinctly.
[3] Available at: `http://imdc.datcat.org/collection/1-003M-5`
[4] Homepage: `http://nasadi.github.io/Zambezi/`

The remainder of the work is organized as follows: Section 2 provides background concepts on real-time index features. Furthermore, we review related work that examine both efficiency and effectiveness; Section 3 describes the employed collection and introduces vocabulary dynamics analysis; Section 4 presents the index entries invalidator approach; Section 5 details the methodology applied during experiments, along with the employed metrics. Also the obtained results are displayed. 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 [25]. 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 [4, 24, 25]. 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 [2].

Nowadays, Twitter's Earlybird retrieval engine [6], 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 [3]. 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. [8] 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 [11] 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, [9, 19] 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. [20] 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

Data Characterization: Tweets2011 [17] is constituted as the reference collection of the TREC-2012 Microblog Track [22]. 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 and it is employed in various works [9, 3, 1]. Due to its sampling methodology, the corpus tends to degrade over time so that 11,601,066 tweets were downloaded successfully. Additionally, all non-English tweets were filtered out. The tweets distribution over time is shown in Figure 1, on average $9,352$ posts arrive per hour. Each post is composed of roughly 13.39 useful words and 81.25 characters. As stated in [20], 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[5],

[5] Tokens that are preceded by '@' symbol are used to refer to user's alias.

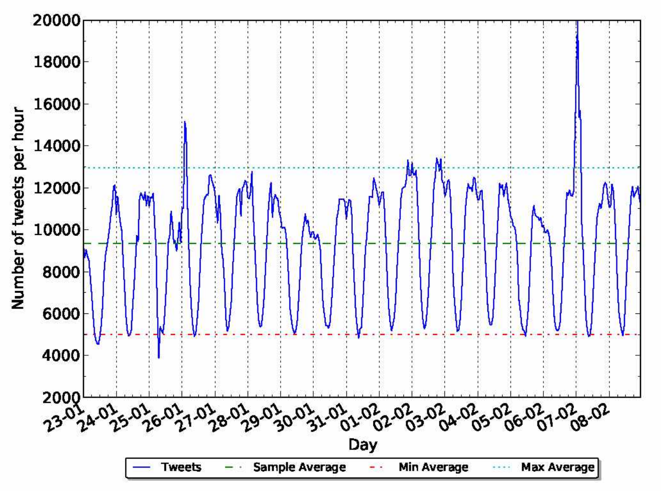


Fig. 1: Tweets arrival rate

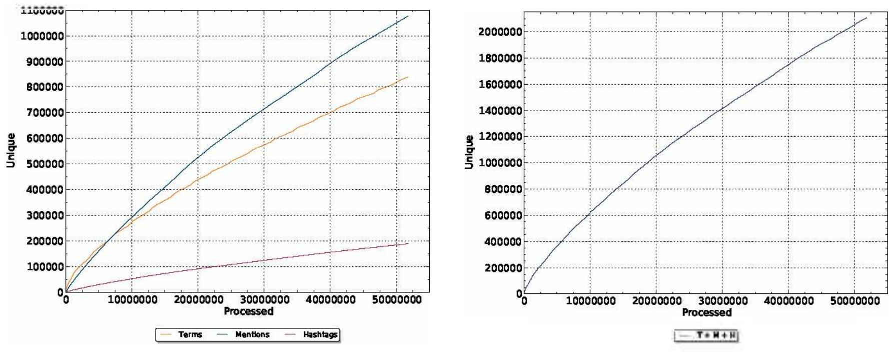
hashtags[6], and general terms. The reason for applying this convention lies behind the fact that both mentions and hashtags have a particular meaning and value inside Twitter [10, 23]. According to traditional IR literature, a practical way to describe how vocabulary and collection size are related corresponds to Heaps' law [16]. In particular, it enables to empirically estimate vocabulary size (and its growth) as a function of the collection size. Figure 2(a) shows that the considered tokens growth faster than this law predictions, and they rather exhibit a linear growth. Similarly, without applying a per token distinction (Figure 2(b)). Despite the number of new encountered tokens raises considerably fast, mostly of those tokens are *hapax*[7], *i.e.*, they appear in only a single tweets. They represent roughly 67.7% of the whole vocabulary.

As various works [9, 19, 20] state, 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.

In order to proceed with vocabulary dynamics characterization, we classify tokens in three groups by studying how its 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 daily a basis. We set a boolean true value if a token is present in at least one the processed documents of the corresponding sample day. Every day, tokens' frequency information is updated according to

[6] Tokens that are preceded by '#' symbol represent a kind of keyword, and usually adopt the *camel case* convention.

[7] A *hapax legomenon*, or just hapax, is a word that occurs only once within a context.



(a) Per considered token type

(b) Whole vocabulary (Without distinction of token type)

Fig. 2: Vocabulary Growth. The X-axis corresponds to the number of processed tokens. The Y-axis shows unique tokens (That compose the vocabulary).

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 with $s = 7$ (a whole week) that we consider as a reasonable number to study how tokens behave, giving the dynamic of Twitter. Figure 3 shows the distribution of the classified tokens across the sample. Note that we are able to start with this labeling on January 28th, because until then the window has not reached the 7th slot. Even though we apply a window of seven slots, a deeper study to determine the impact of the window size in this token classification is required.

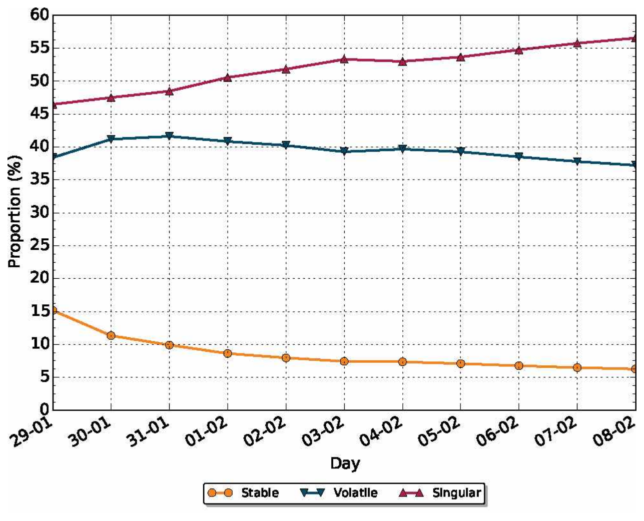


Fig. 3: Proportion of each type of token over days

4 Index Entries Invalidator

According to the preceding analysis it is reasonable to think about pruning certain 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 [15] 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 a time-to-live (TTL) strategy, as adopted in result caches [7]. The IEI is based on the time that entries have persisted in the index without been updated. When a token exceeds a certain 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 time it was updated is larger than the TTL value.

In order to evaluate this approach in a real time search scenario, we modify the Zambezi search engine, whose source code is publicly available. We implement the index entries invalidator to perform different experiments to ascertain the efficiency and effectiveness of the proposed approach.

5 Experimental Evaluation and Results

Methodology: Initially we set the TTL 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[8] 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 the MBWAND algorithm (basically, it uses timestamps as sorting criteria) that is suitable for microblogging documents. In other words, the answer set comprises a list of relevant tweets ordered from newest to oldest. Finally, to ensure the consistency of the results, we perform five trials of the experiment, and average the outcomes.

Due to lack of a publicly available real-time query set various works build up synthetic ones. Initially, we evaluate our approach with a shred of the synthetic query log generated in [20]. However, the number of queries in the set is not enough to run a robust performance evaluation. For this reason, we decide to use the one million queries extracted from the well-known AOL Query Log [21], employed in other works involving real-time search [15, 3].

Metrics: To assess the overall performance of the approach we evaluate both efficiency (time and space) and 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 approach. 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) [18]. In order to accomplished this, the retrieved tweets are brought to the user in reverse chronological sorting.

Results: The effectiveness evaluation shows that the IEI doesn't 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, our approach reduces the overall execution time in all configurations up to 6% in the best case. Figure 5 shows these results. The increased execution time over days corresponds to the vocabulary growth. The difference also increases conforming more tokens are added to the baseline thus, more tokens are invalidated and pruned. This suggests that a more aggressive pruning strategy is needed to complement the IEI and to control the vocabulary growth, in particular, posting lists pruning may be adequate (this is a direction for future research).

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.

[8] Our sample preserves the original query length distribution.

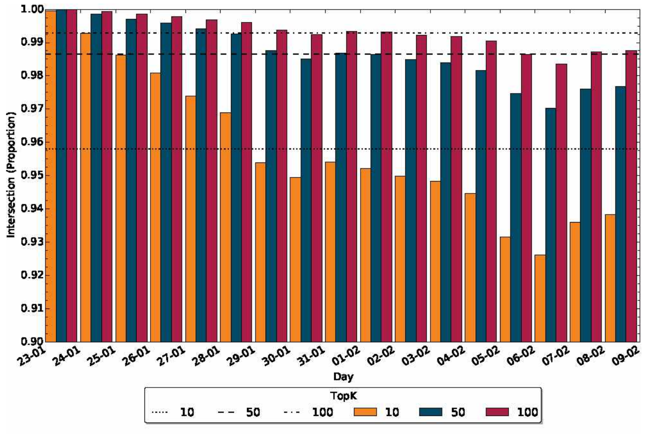


Fig. 4: Intersection proportion when retrieving top 10, 50 and 100 documents. Dotted lines correspond to mean values for each series.

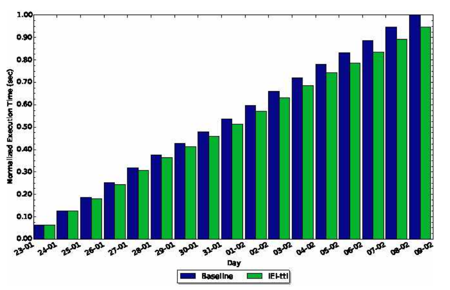
The number of valid entries decreases dramatically over days (up to 88%). This enables faster lookups into the vocabulary. However, the total number of DocIDs that accounts for the resulting posting lists decreases much slower (Table 2), up to 9.1% in the last day. Again, this happens due to the pruned tokens correspond to infrequent ones whose document frequency is very low (typically, one). This result reinforces the idea 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.

Table 1: Number of entries in the inverted index

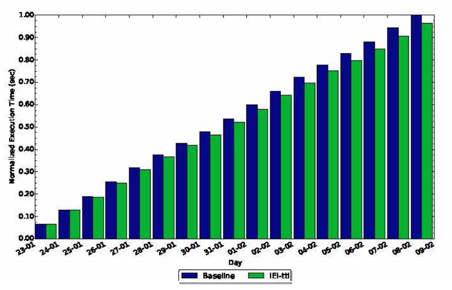
Day	**Baseline**	**Pruned**	**Diff. %**
1	217,044	209,379	3.53
2	390,863	231,976	40.65
3	540,456	238,330	55.90
4	692,677	259,240	62.57
5	833,499	250,156	69.99
6	970,939	256,852	73.55
7	1,081,271	223,664	79.31
8	1,186,153	228,312	80.75
9	1,295,470	239,079	81.54
10	1,407,580	258,560	81.63
11	1,519,236	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 2: Sum of DocIDs in the posting lists of all terms

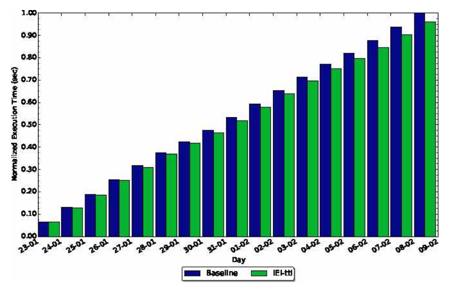
Day	**Baseline**	**Pruned**	**Diff. %**
1	2,392,099	2,384,262	0.33
2	5,174,584	4,992,039	3.53
3	7,864,478	7,467,306	5.05
4	10,919,587	10,294,900	5.72
5	13,892,852	12,991,206	6.49
6	16,859,239	15,678,477	7.00
7	19,413,681	17,914,817	7.72
8	22,037,617	20,270,901	8.02
9	24,934,503	22,903,479	8.15
10	27,976,632	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



(a) Top-10 results



(b) Top-50 results



(c) Top-100 results

Fig. 5: Normalized Execution Time for 1.000.000 queries

6 Conclusion and Future Work

In this work we introduced the concept of index entries invalidator, an approach that aims to 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. Our experimental results showed that the proposed approach reduce the number of entries in the vocabulary up to an 88%, enabling faster lookups. The overall execution time for our query set is also reduced up to 6%.

However, the resulting size of the index decreases much slower, up to 9.1%. 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, will be necessary to consider the posting list lengths of the tokens that remain into the vocabulary and study how they evolve over time. Moreover, we are interested in defining a second kind of invalidator following a different view of the vocabulary dynamics.

Acknowledgement

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. Asadi, N., Lin, J.: Fast candidate generation for real-time tweet search with bloom filter chains. ACM Trans. Inf. Syst. 31(3), 13:1–13:36 (2013)
2. Asadi, N., Lin, J.: An exploration of postings list contiguity in main-memory incremental indexing. LSDS-IR '14, New York, NY, USA (2014)
3. Asadi, N., Lin, J., Busch, M.: 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. pp. 1186–1194. KDD '13, ACM, New York, NY, USA (2013)
4. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval - the concepts and technology behind search, Second edition. Pearson Education Ltd., Harlow, England (2011)
5. Blanco, R., Bortnikov, E., Junqueira, F., Lempel, R., Telloli, L., Zaragoza, H.: Caching search engine results over incremental indices. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 82–89. SIGIR '10, ACM, New York, NY, USA (2010)
6. Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S., Lin, J.: Earlybird: Real-time search at twitter. In: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering. pp. 1360–1369. ICDE '12, IEEE Computer Society, Washington, DC, USA (2012)
7. Cambazoglu, B.B., Junqueira, F.P., Plachouras, V., Banachowski, S., Cui, B., Lim, S., Bridge, B.: A refreshing perspective of search engine caching. In: Proceedings of the 19th International Conference on World Wide Web. pp. 181–190. WWW '10, ACM, New York, NY, USA (2010)
8. Chen, C., Li, F., Ooi, B.C., Wu, S.: Ti: An efficient indexing mechanism for real-time search on tweets. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. pp. 649–660. SIGMOD '11, ACM, New York, NY, USA (2011)
9. Choi, J., Croft, W.B.: Temporal models for microblogs. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management. pp. 2491–2494. CIKM '12, ACM, New York, NY, USA (2012)
10. Efron, M.: Information search and retrieval in microblogs. J. Am. Soc. Inf. Sci. Technol. 62(6), 996–1008 (2011)
11. Grant, C.E., George, C.P., Jenneisch, C., Wilson, J.N.: 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)
12. Huberman, B.A., Romero, D.M., Wu, F.: Social networks that matter: Twitter under the microscope. CoRR abs/0812.1045 (2008)
13. Java, A., Song, X., Finin, T., Tseng, B.: 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. pp. 56–65. WebKDD/SNA-KDD '07, ACM, New York, NY, USA (2007)

14. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web. pp. 591–600. WWW '10, ACM, New York, NY, USA (2010)
15. Lin, J., Mishne, G.: 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)
16. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2008)
17. McCreadie, R., Soboroff, I., Lin, J., Macdonald, C., Ounis, I., McCullough, D.: On building a reusable twitter corpus. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1113–1114. SIGIR '12, ACM, New York, NY, USA (2012)
18. McCullough, D., Lin, J., Macdonald, C., Ounis, I., McCreadie, R.: 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)
19. Metzler, D., Cai, C., Hovy, E.: 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. pp. 646–655. NAACL HLT '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012)
20. Nepomnyachiy, S., Gelley, B., Jiang, W., Minkus, T.: What, where, and when: Keyword search with spatio-temporal ranges. In: Proceedings of the 8th Workshop on Geographic Information Retrieval. pp. 2:1–2:8. GIR '14, ACM, New York, NY, USA (2014)
21. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: Proceedings of the 1st International Conference on Scalable Information Systems. InfoScale '06, ACM, New York, NY, USA (2006)
22. Soboroff, I., Ounis, I., Macdonald, C., Lin, J.: Overview of the trec-2012 microblog track. In: In Proceedings of TREC 2012 (2012)
23. Teevan, J., Ramage, D., Morris, M.R.: #twittersearch: A comparison of microblog search and web search. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining. pp. 35–44. WSDM '11, ACM, New York, NY, USA (2011)
24. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes (2Nd Ed.): Compressing and Indexing Documents and Images. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
25. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. 38(2) (2006)