Tracking unbounded Topic Streams

Dominik Wurzer
School of Informatics
University of Edinburgh
d.s.wurzer
@sms.ed.ac.uk

Victor Lavrenko
School of Informatics
University of Edinburgh
vlavrenk
@inf.ed.ac.uk

Miles Osborne
Bloomberg
London
mosborne29
@bloomberg.net

Abstract

Tracking topics on social media streams is non-trivial as the number of topics mentioned grows without bound. This complexity is compounded when we want to track such topics against other fast moving streams. We go beyond traditional small scale topic tracking and consider a stream of topics against another document stream. We introduce two tracking approaches which are fully applicable to true streaming environments. When tracking 4.4 million topics against 52 million documents in constant time and space, we demonstrate that counter to expectations, simple single-pass clustering can outperform locality sensitive hashing for nearest neighbour search on streams.

1 Introduction

The emergence of massive social media streams has sparked a growing need for systems able to process them. While previous research (Hassan et al., 2009; Becker et al., 2009; Petrovic et al., 2010; Cataldi et al., (2010); Weng et al., (2011); Petrovic 2013) has focused on detecting new topics in unbounded textual streams, less attention was paid to following (tracking) the steadily growing set of topics. Standard topic tracking (Allan, 2002) deals with helping human analysts follow and monitor ongoing events on massive data streams. By pairing topics with relevant documents, topic tracking splits a noisy stream of documents into sub-streams grouped by their target topics. This is a crucial task for financial and security analysts who are interested in pulling together relevant information from unstructured and noisy data streams. Other fields like summarization or topic modeling benefit from topic tracking as a mean to generate their data sources.

In todays data streams however, new topics emerge on a continual basis and we are interested in following all instead of just a small fraction of newly detected topics. Since its introduction (Allan, 2002), standard topic tracking typically operates on a small scale and against a static set of predefined target topics. We go beyond such approaches and deal for the first time with massive, unbounded topic streams. Examples of unbounded topic streams include all events reported by news agencies each day across the world; popular examples of unbounded document streams include social media services such as Twitter. Tracking streams of topics allows research tasks like topic-modeling or summarization to be applied to millions of topics, a scale that is several orders of magnitude larger than those of current publications. We present two massive scale topic tracking systems capable of tracking unbounded topic streams. One is based on locality sensitive hashing (LSH) and the other on clustering. Since we operate on two unbounded data sources we are subject to the streaming model of computation (Muthukrishnan, 2005), which requires instant and single-pass decision making in constant time and space. Contrary to expectations, we find that nearest neighbour search on a stream based on clustering performs faster than LSH for the same level of accuracy. This is surprising as LSH is widely believed to be the fastest way of nearest neighbour search. Our experiments reveal how simple single-pass clustering outperforms LSH in terms of effectiveness and efficiency. Our results are general and apply to any setting where we have massive or infinite numbers of topics, matched against unboundedly large document streams.

Contributions

- For the first time we show how it is possible to track an unbounded stream of topics in constant time and space, while maintaining a level of effectiveness that is statistically indistinguishable from an exact tracking system
- We show how single-pass clustering can outperform locality sensitive hashing in terms of effectiveness and efficiency for identifying nearest neighbours in a stream
- We demonstrate that standard measures of similarity are sub-optimal when matching short documents against long documents

2 Related Work

Topic or event tracking was first introduced in the Topic Detection and Tracking (TDT) program (Allan, 2002). In TDT, topic tracking involves monitoring a stream of news documents with the intent to identify those documents relevant to a small predefined set of target topics. During the course of TDT, research focused extensively on the effectiveness of tracking systems, neglecting scale and efficiency. The three official data sets only range from 25k to 74k documents with a few hundred topics (Allan, 2002).

More recently, the rise of publicly available real-time social media streams triggered new research on topic detection and tracking, intended to apply the technology to those high volume document streams. The novel data streams differ from the TDT data sets in their volume and level of noise. To provide real-time applications, traditional methods need to be overhauled to keep computation feasible. It became common practice to limit data sets to cope with the computational effort. Popular strategies involve reducing the number of tracked topics (Lin et al., 2011; Nichols et al., 2012;) as well as sampling the document stream (Ghosh et al., 2013). These approaches have proven to be efficient in cutting down workload but they also limit an application's performance. Furthermore, Sayyadi et al. (2009) discovered and tracked topics in social streams based on keyword graphs. They applied the sliding window principle to keep the computation feasible, although their data set only contained 18k documents. Yang et al. 2012 tracked topics in tweet streams using language models. To cope with the computational effort they assume a small set of topics of only a few dozen, which are defined in advance. Tang et al. (2011) tracked a single topic on a few thousand blogs based on semantic graph topic models. Pon et al. (2007) recommend news by tracking multiple topics for a user but their data sets only span several thousand documents and a few topics.

Further related work includes the real-time filtering task, introduced as part of TREC's Microblog Track in 2012 (Soboroff et al., 2012). Hong et al. (2013) explore topic tracking in tweet streams in relation to the TREC real-time filtering task by relying on a sliding window principle, while focusing on the cold start problem.

3 Topic Tracking

3.1 Traditional Approach

Numerous approaches to topic tracking have emerged, spanning from probabilistic retrieval to statistical classification frameworks. While there is no single general approach, we define the traditional approach to tracking from a high-level perspective covering the basic principle of all previous approaches. We do not make any assumptions about the kind of topics, documents or distance functions used. As defined by TDT (Allan, 2002), we assume, we operate on an unbounded document stream with the goal of tracking a fixed set of target topics. Although topics are allowed to drift conceptually and evolve over time, new topics would always trigger the start of a new tracking system.

Algorithm 1 Traditional Tracking

```
INPUT:
TOPIC-SET $\{t \in T\}$
DOCUMENT-STREAM $\{d \in D\}$
OUTPUT:
relevant topic-document pairs $\{t, d\}$

while documents d in stream D do
    for all topics t in set T do
        similarity = computeSimilarity(d,t)
        if similarity > threshold then
            emit relevant $\{t, d\}$
```

As seen in Algorithm 1, documents arrive one at a time, requiring instant decision making through single pass processing. Each document is compared to all topics representations to identify the closest topic. The tracking decision is based on the similarity to the closest topic and usually defined by a thresholding strategy. Because incoming documents can be relevant to more than one topic, we need to match it against all of them. Due to its simplicity, the traditional tracking approach is highly efficient when applied to a fairly low number of topics.

3.2 Shortcomings of the traditional approach

The traditional approach - though low in computational effort - becomes challenging when scaling up the number of target topics. The computational effort arises from the number of comparisons made (the number of documents times topics). That explains, why researches following the traditional approach have either lowered the number of documents or topics. Heuristics and indexing methods increase the performance but offer no solution scalable to true streaming environments because they only allow for one-side scaling (either a large number of documents or topics). Increasing either of the two components by a single document, increases the computational effort by the magnitude of the other one. For the extreme case of pushing to an infinite number of topics, tracking in constant space is a necessity.

4 Tracking at scale

Before directly turning to a full streaming set up in constant space, we approach tracking a topic stream on a document stream in unbounded space. The key to scale up documents and topics, lies in reducing the number of necessary comparisons. Throughout the remainder of this paper we represent documents and topics arriving from a steady high volume stream by term-weighted vectors in the vector space.
In order to cut down the search space, we encapsulate every topic vector by a hypothetical region marking its area of proximity. Those regions are intend to capture documents that are more likely to be relevant. Ideally, these regions form a hypersphere centred around every topic vector with a radius equal to the maximum distance to relevant documents. The tracking procedure is then reduced to determining whether an incoming document is also enclosed by any of the hyperspheres.

4.1 Approximated Tracking

Our first attempt to reach sub-linear execution time uses random segmentation of the vector space using hashing techniques. We frame the tracking process as a nearest neighbour search problem, as defined by Gionis et al. (1999). Documents arriving from a stream are seen as queries and the closest topics are the nearest neighbours to be identified. We explore locality sensitive hashing (LSH), as described by Indyk et al. (1998), to approach high dimensional nearest neighbour search for topic tracking in sub-linear time. LSH, which has been used to speed up NLP applications (Ravichandran et al., 2005), provides hash functions that guarantee that similar documents are more likely to be hashed to the same binary hash key than distant ones. Hash functions capture similarities between vectors in high dimensions and represent them on a low dimensional binary level. We apply the scheme by Charikar (2002), which describes the probabilistic bounds for the cosine similarity between two vectors. Each bit in a hash key represents a documents position with respect to a randomly placed hyperplane. Those planes segment the vector space, forming high dimensional polygon shaped buckets. Documents and topics are placed into a bucket by determining on which side of each the hyperplanes they are positioned. We interpret these buckets as regions of proximity as the collision probability is directly proportional to the cosine similarity between two vectors.

Algorithm 2 LSH-based Tracking

```
INPUT:
TOPIC-STREAM $\{T\}$
DOCUMENT-STREAM $\{D\}$
OUTPUT:
relevant topic-document pairs $\{t, d\}$

while document d in T, D do
    if d $\epsilon$ T then
        hashKeys = $hash_{LSH}$(d)
        store hashKeys in hashTables
    else if d $\epsilon$ D then
        candidateSet = lookupHashtables($hash_{LSH}$(d))
        for all topics t in candidateSet do
            if similarity(d,t) $>$ threshold then
                emit relevant $\{t, d\}$
```

Algorithm 2 outlines the pseudo code to LSH-based tracking. Whenever a topic arrives, it is hashed, placing it into a bucket. To increase collision probability with similar documents, we repeat the hashing process with different hash functions, storing a topic and hash-key tuple in a hash table. On each arrival of a new document the same hash functions are applied and the key is matched against the hash tables, yielding a set of candidate topics. The probabilistic bounds of the hashing scheme guarantee that topics in the candidate set are on average more likely to be similar to the document than others.

We then match each topic in the candidate set against the document to lower the false positive rate of LSH (Gionis, et al., 1999). The number of exact comparisons necessary is reduced to the number of topics in the candidate set.

4.2 Cluster based Tracking

LSH based tracking segments the vector-space randomly without consideration of the data's distribution. In contrast, we now propose a data dependent approach through document clustering. The main motivation for data dependent space segmentation is increased effectiveness resulting from taking the topic distribution within the vector space into account when forming the regions of proximity. We construct these regions by grouping similar topics to form clusters represented by a centroid. When tracking a document, it is matched against the centroids instead of all topics, yielding a set of candidate topics. This allows reducing the number of comparisons necessary to only the number of centroids plus the number of topics captured by the closest cluster.

Algorithm 3 Cluster based Tracking

```
INPUT:
INITIAL-CLUSTER-SET $\{c \epsilon C\}$
TOPIC-STREAM $\{T\}$
DOCUMENT-STREAM $\{D\}$
threshold for spawning a new cluster $\{thr_{spawn}\}$
threshold for adapting an existing cluster $\{thr_{adapt}\}$
OUTPUT:
relevant topic-document pairs $\{t, d\}$

while document d in T, D do
    if d $\epsilon$ T then
        $c_{min} = argmin_c\{distances(d, c \epsilon C)\}$
        if distance(d,$c_{min}$) $> thr_{spawn}$ then
            spawnNewCluster(d $\rightarrow$ C)
        else if distance(d,$c_{min}$) $< thr_{adapt}$ then
            contribute,assign($c_{min}$,d)
        else
            assign($c_{min}$,d)
    else if d $\epsilon$ D then
        $c_{min} = argmin_c\{$distances(d,c $\epsilon C)\}$
        candidateSet = $\{t \epsilon c_{min}\}$
        for all topics t in candidateSet do
            if similarity(d,t) $>$ threshold then
                emit relevant $\{t, d\}$
```

While the literature provides a vast diversity of clustering methods for textual documents, our requirements regarding tracking streams of topics naturally reduce the selection to lightweight single-pass algorithms. Yang et al. (2012) provided evidence that in extreme settings simple approaches work well in terms of balancing effectiveness, efficiency and scalability. We identified ArteCM by Carullo et al. (2008), originally intended to cluster documents for the web, as suitable. Algorithm 3 outlines our approach for cluster based tracking. Given an initial set of 4 random centroids, we compare each arriving topic to all centroids. We associate the new topic with the cluster whenever it is close enough. Particularly close documents contribute to a cluster, allowing it to drift towards topic dense regions. If the document is distant to all existing clusters, we spawn a new cluster based on the document.

Documents arriving from the document stream are exactly matched against all centroids to determine the k-closest clusters. Topics associated with those clusters are subsequently exhaustively compared with the document, yielding topic-document pairs considered to be relevant. Probing more than one cluster increases the probability of finding similar topics. This does not correlate with soft-clustering methods as multiple probing happens at querying time while topics are assigned under a hard clustering paradigm.

4.3 Algorithm Comparison

Both the LSH- and the cluster-based tracking algorithm provide two parameters that are conceptually directly comparable to each other. The number of bits per hash key and the threshold for spawning new clusters directly determine the size of the candidate set by either varying the bucket size or the cluster radius. The size of the candidate set trades a gain in efficiency against a loss in effectiveness. Fewer topics in the candidate set heavily reduce the search space for the tracking process but increase the chance of missing a relevant topic. Bigger sets are more likely to cover relevant topics but require more computational effort during the exact comparison step. The proposed algorithms allow continuously adjusting the candidate set size between two extremes of having all topics in a single set and having a separate set for each topic.

The second parameter both algorithms have in common, is the number of probes to increase the probability of identifying similar topics. While LSH-based tracking offers the number of hash tables, cluster-based tracking provides the number of clusters probed. We again encounter a trade-off between gains in efficiency at the cost of effectiveness. Each additionally probed cluster or looked up table increases the chance of finding relevant topics as well as the computational effort.

5 Tracking Streams in Constant Space

Operation in constant space is crucial when tracking topic streams. We ensure this by placing an upper limit on the number of concurrently tracked topics. Whenever the limit is reached, an active topic is deleted and subsequently not considered any longer. The strategy for selecting deletion candidates is heavily application dependant. To handle topic streams, LSH-based tracking replaces the entries of an active topic in its hash-tables by the values of the new topic, whenever the maximum number of topics is reached. Cluster-based tracking requires more adaptation because we allow clusters to drift conceptually. Whenever the maximum number of topics is reached, the contribution of the deletion candidate to its cluster is reverted and it is removed, freeing space for a new topic.

6 Experiments

We evaluate the three algorithms in terms of effectiveness and efficiency. Starting out with tracking a small set of topics using the traditional approach, we evaluate various similarity metrics to ensure high effectiveness. We then conduct scaling experiments on massive streams in bounded and unbounded space.

Corpora

Traditional tracking datasets are unsuitable to approach tracking at scale as they consist of only a few thousand documents and several hundred topics (Allan, 2002). We created a new data set consisting of two streams (document and topic stream). The document stream consists of 52 million tweets gathered through Twitter's streaming API [1]. The tweets are order by their time-stamps. Since we are advocating a high volume topic stream, we require millions of topics. To ensure a high number of topics, we treat the entire English part (4.4 mio articles) of Wikipedia[2] as a proxy for a collection of topics and turn it into a stream. Each article is considered to be an unstructured textual representation of a topic time-stamped by its latest verified update.

[1]http://stream.twitter.com

[2]http://en.wikipedia.org/wiki/Wikipedia_database

Relevance Judgements

The topics we picked range from natural disasters, political and financial events to news about celebrities, as seen in table 3. We adopted the search-guided-annotation process used by NIST (Fiscus et al., 2002) and followed NIST's TDT annotation guidelines. According to the definition of TDT, a document is relevant to a topic if it speaks about it (Allan, 2002). In total we identified 14,436 tweets as relevant to one of 30 topics.

total number of topics	4.4 mio
annotated topics	30
total number of documents	52 mio
documents relevant to one of the 30 annotated topics	14.5k

Table 1: Data set statistics

Baseline

We use an exact tracking system as a baseline. To speed up runtime, we implement an inverted index in conjunction with term-at-a-time query execution. Additionally, we provide a trade off between effectiveness and efficiency by randomly down sampling the Twitter stream. Note that this closely resembles previous approaches to scale topic tracking (Ghosh et al., 2013).

Evaluation Metrics

We evaluate effectiveness by recall and precision and combine them using F1 scores. Efficiency is evaluated using two different metrics. We provide a theoretical upper bound by computing the number of dot products required for tracking (Equations 1-4).

$$DP_{traditional} = n_D * n_T \quad (1)$$

$$DP_{LSH-based} = (n_D + n_T) * (k * L) + DP_{cs} \quad (2)$$

$$DP_{cluster-based} = (n_D + n_T) * c + DP_{cs} \quad (3)$$

$$DP_{cs} = n_D * n_C \quad (4)$$

Variables	**Definition**
n_D	total number of documents
n_T	total number of topics
k	number of bits per hash
L	total number of hash tables
c	total number of clusters
n_C	total number of topics in all candidate sets

Table 2: Definition of variables for equation 1-4

Topic-Title	Topic description	Number of relevant tweets
Amy Winehouse	Amy Winehouse dies	3265
Prince William	William and Kate arrive in Canada	1021
Floods in Seoul	Floods and landslides in North and South Korea	432
Flight 4896	Flight 4896 crashed	11
Bangladesh-India border	Bangladesh and India sign a border pact	4
Goran Hadzic	War criminal Goran Hadzic got arrested	2

Table 3: Showing 6 example topics plus a short summary of relevant tweets, as well as the number of relevant tweets per topic

They therefore indicate performance without system- or implementation-dependent distortions. Equations 2 and 3 represent the cost to identify the candidate set for the LSH- and cluster-based algorithm plus the cost resulting from exhaustively comparing the candidate sets with the documents (Equation 4).

Because we compute the dot products for a worst case scenario, we also provide the runtime in seconds. All run-times are averaged over 5 runs, measured on the same idle machine. To ensure fair comparison, all algorithms are implemented in C using the same libraries, compiler, compiler optimizations and run as a single process using 4 GB of memory. Because the runtime of the traditional approach ($\sim$171 days) exceeds our limits, we estimate it based on extrapolating 50 runs using up to 25,000 topics. Note that this extrapolation favours the efficiency of the baseline system as it ignores hardware dependent slowdowns when scaling up the number of topics.

6.1 Exact tracking

In our first experiment we track 30 annotated topics on 52 million tweets using the traditional approach. We compare various similarity measures (Table 4) and use the best-performing one in all following experiments. Our data set differs from the TREC and TDT corpora, which used news-wire articles. Allan et al. (2000) report that the cosine similarity constantly performed as the best distance function for TDT. The use of Wikipedia and Twitter causes a different set of similarity measures to perform best. This results from the imbalance in average document length between Wikipedia articles (590 terms) and tweets (11 terms). The term weights in short tweets (many only containing a single term) are inflated by the cosine's length normalization. Those short tweets are however not uniquely linkable to target topics and consequently regarded as non-relevant by annotators, which explains the drop in performance. The similarity function chosen for all subsequent experiments is a BM25 weighted dot product, which we found to perform best.

	F1 score
tf-idf weighted cosine	0.147
tf-idf weighted dot product	0.149
BM25 weighted cosine	0.208
BM25 weighted dot product	0.217

Table 4: Comparing the effectiveness of similarity measures when matching 30 Wikipedia articles against 52 million tweets

6.2 Tracking at scale, using Wikipedia and Twitter

Previously, we conducted small scale experiments, now we are looking to scale them up, by tracking 4.4 million Wikipedia articles on 52 million tweets without limiting the number of topics tracked. The resulting trade-off between effectiveness and efficiency is shown in Figure 1 and 2. The right-most point corresponds to exhaustive comparison of every document against every topic – this results in highest possible effectiveness (F1 score) and highest computational cost. All runs use optimal tracking thresholds determined by sweeping them while optimizing on F1 score as an objective function. We also show the performance resulting from the traditional approach when randomly down-sampling the document (Twitter) stream, which resembles previous attempts to scale tracking (Ghosh et al., 2013). Every point on the LSH-based tracking curve in Figure 1 and 2 represents a different number of bits per hash key (varying between 4 and 20) and tables (ranging from 6 to 200). The points on the cluster-based tracking curves result from varying the number of clusters (ranging from 1 to 100,000) and probes. The resulting bucket sizes span from a few dozen to over a million topics.

As expected, the graphs in Figure 1 closely resembles those in Figure 2. The two figures also show that the performance of all three algorithms is continuously adjustable. Unsurprisingly, LSH- and cluster-based tracking clearly outperform

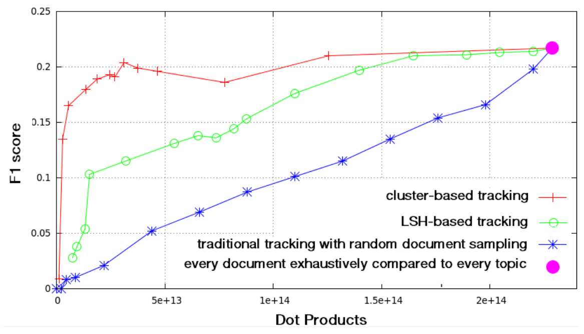

Figure 1: Trade-off between efficiency and dot-products for LSH- and cluster-based tracking as well as a random down-sampling approach for traditional tracking

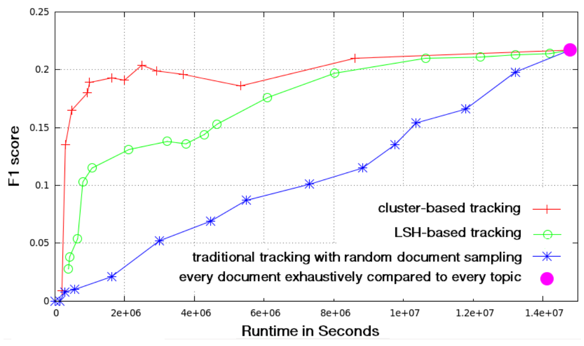

Figure 2: Trade-off between efficiency and runtime for LSH- and cluster-based tracking as well as a random down-sampling approach for traditional tracking;

random document sampling for the traditional approach, based on their more effective search space reduction strategies. More surprisingly, we also observe that cluster-based tracking outperforms tracking based on LSH in terms of efficiency for F1 scores between 10% and 20%. To understand why tracking based on clustering is faster than randomized tracking, we further investigate their abilities in cutting down the search space.

Figure 3 presents the candidate set size necessary to find a certain ratio of relevant topics. The graph also illustrates the impact of probing multiple clusters. When focusing on a recall up to 60%, LSH-based tracking requires a significantly larger candidate set size in comparison with tracking through clustering. For example, LSH-based tracking needs to examine 30% of all topics to reach a recall of 50%, while the cluster based approach only needs to look at 9%. This effect diminishes for higher recall values. Furthermore, we observe an impressive performance gain in recall from 20% to 60%, resulting from additionally probing the k-closest clusters instead

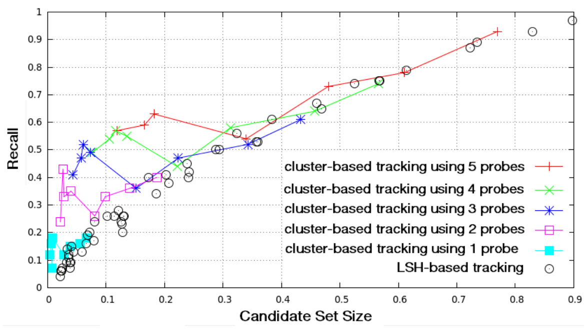

Figure 3: Comparing the candidate set size with the Recall of LSH- with cluster-based tracking without the exact evaluation phase; The magnitude of the candidate set size represents the ratio between the number of candidate topics and the total number of topics;

of just the closest one. While data dependent segmentation is expected to outperform LSH in terms of effectiveness, we were surprised by the magnitude of its impact on efficiency.

The lack in effectiveness of LSH has a direct negative implication on its efficiency for tracking. In order to make up for its suboptimal space segmentation, it requires substantially bigger candidate sets to reach the same level of recall as the cluster-based approach. The size of the candidate set is critical because we assume a subsequent exact comparison phase to lower the false positive rate. The overhead of both algorithms is outweighed by the cost of exact comparison for the candidate set.

Table 5, which compares the performance of the three algorithms, reveals a drastic reduction in runtime of up to 80%, at the cost of only a minor decrease in F1 score. The differences of 6% and 10% percent in F1 score are statistically not significant according to a sign test ($p<=0.362$ and $p<=0.2$). Consequently, both algorithms achieve substantial runtime reduction, while maintaining a level of effectiveness that is statistically indistinguishable from the traditional (exact) approach.

6.3 Tracking Wikipedia on Twitter in constant space

Tracking a stream of topics in bounded space is highly application specific due to the deletion procedure. We know from previous studies (Nichols et al., 2012) that a topic's popularity within Twitter fades away over time. We are interested in keeping currently active topics and delete those that attract the least number of recent documents. This set-up has the interesting aspect that the doc-

Algorithm	F1 score	Dot Products	Runtime (sec)
traditional approach	0.217	$2.3 * 10^{14}$	$1.5 * 10^{7}$
LSH-based tracking	0.196 (-10%)	$1.4 * 10^{14}$ (-39%)	$8.0 * 10^{6}$ (-46%)
cluster-based tracking	0.204 (-6%)	$3.1 * 10^{13}$ (-86%)	$2.5 * 10^{6}$ (-83%)

Table 5: Effectiveness and efficiency of LSH- and cluster-based tracking to the traditional approach

Algorithm	Space	F1 score	dot products	runtime (sec)
LSH-based tracking	unbounded	0.196	$1.4 * 10^{14}$	$8.0 * 10^{6}$
	bounded	0.173 (-12%)	$5.1 * 10^{11}$ (-99%)	$4.1 * 10^{4}$ (-99%)
cluster-based tracking	unbounded	0.204	$3.1 * 10^{13}$	$2.5 * 10^{6}$
	bounded	0.189 (-7%)	$1.8 * 10^{11}$ (-99%)	$3.3 * 10^{4}$ (-98%)

Table 6: Effectiveness and efficiency for tracking in bounded and unbounded space

ument stream dictates the lifespan of each topic in the topic stream. Table 6 contains the results of cluster- and LSH-based tracking and compares them to their bounded versions using the same set up. Note that the hit in performance is solely defined by the amount of memory provided and therefore continuously adjustable.
For this particular experiment, we chose an upper bound of 25k concurrent topics. The table represents a substantial drop in runtime, following the reduced search space, at a fairly low expense in effectiveness. Based on our observations, we hypothesise that significant topics are more likely to be discussed during random Twitter chatter than the average Wikipedia topic. It is interesting to notice that the runtime also indicates a lower overhead for LSH-based tracking in comparison with the cluster-based approach. This difference was hidden in the unbounded tracking experiments but carries now more weight.

7 Conclusion

We extended traditional topic tracking by demonstrating that it is possible to track an unbounded stream of topics in constant space and time. We also presented two approaches to tracking, based on LSH and clustering that efficiently scale to a high number of topics and documents while maintaining a level of effectiveness that is statistically indistinguishable from an exact tracking system. While they trade gains in efficiency against a loss in effectiveness, we showed that cluster based tracking does so more efficiently due to more effective space segmentation, which allows a higher reduction of the search space. Contrary to common believes this showed how nearest neighbour search in data streams based on clustering performs faster than LSH, for the same level of accuracy. Furthermore, we showed that standard measures of similarity (cosine) are sub-optimal when tracking Wikipedia against Twitter.

References

James Allan, Victor Lavrenko, Daniella Malin, and Russell Swan. 2000. Detections, bounds, and timelines: Umass and tdt-3. In Proceedings of Topic Detection and Tracking Workshop, pages 167-174.

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '98). ACM, New York, NY, USA.

James Allan. 2002. Topic Detection and Tracking: Event-Based Information Organization. Kluwer Academic Publishers, Norwell, MA, USA.

Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. 2010. Emerging topic detection on Twitter based on temporal and social terms evaluation. In Proceedings of the Tenth International Workshop on Multimedia Data Mining, pages 1-10. ACM.

H. Becker, M. Naaman, and L. Gravano. 2009. Event Identification in Social Media. In 12th International Workshop on the Web and Databases (WebDB'09), Providence, USA.

Moreno Carullo, Elisabetta Binaghi, Ignazio Gallo and Nicola Lamberti. 2008. "Clustering of short commercial documents for the web." Paper presented at the meeting of the ICPR.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In Proceedings of the thirty-fourth annual ACM symposium on Theory of computing (STOC '02). ACM, New York, NY, USA.

Eichmann, D. and P. Sirivasan. 1999. "Filters, Webs and Answers: The University of Iowa TREC-8 Results" Eighth Conference on Text Retrieval, NIST, USA.

Fiscus, J. G. and Doddington, G. R. 2002. Topic detection and tracking evaluation overview. Topic detection and tracking: event-based information organization, pages 17-31.

Saptarshi Ghosh, Muhammad Bilal Zafar, Parantapa Bhattacharya, Naveen Sharma, Niloy Ganguly, and Krishna Gummadi. 2013. On sampling the wisdom of crowds: random vs. expert sampling of the twitter stream. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management (CIKM-13). New York, NY, USA.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. InProceedings of the 25th International Conference on Very Large Data Bases (VLDB '99), San Francisco, CA, USA.

Sayyadi Hassan, Hurst Matthew and Maykov Alexey. 2009. ”Event Detection and Tracking in Social Streams.” In Proceedings of the ICWSM, CA, USA.

Yihong Hong, Yue Fei, and Jianwu Yang. 2013. Exploiting topic tracking in real-time tweet streams. In Proceedings of the 2013 international workshop on Mining unstructured big data using natural language processing. ACM, New York, NY, USA.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbours: towards removing the curse of dimensionality. In Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98). ACM, New York, NY, USA.

Jimmy Lin, Rion Snow, and William Morgan. 2011. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). ACM, New York, NY, USA, 422-429.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. Now Publishers Inc.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. InProceedings of the 2012 ACM international conference on Intelligent User Interfaces (IUI '12). ACM, New York, NY, USA.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to Twitter. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT '10). Association for Computational Linguistics, Stroudsburg, PA, USA.

Sasa Petrovic. 2013. Real-time event detection in massive streams. Ph.D. thesis, School of Informatics, University of Edinburgh.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In Proceedings of ACL.

Raymond K. Pon, Alfonso F. Cardenas, David Buttler, and Terence Critchlow. 2007. Tracking multiple topics for finding interesting articles. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). ACM, New York, NY, USA.

I. Soboroff, I. Ounis, and J. Lin. 2012. Overview of the trec-2012 microblog track. In Proceedings of TREC.

Jintao Tang, Ting Wang, Qin Lu, Ji Wang, and Wenjie Li. 2011. A Wikipedia based semantic graph model for topic tracking in blogosphere. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Three (IJCAI'11).

TDT by NIST - 1998-2004. http://www.itl.nist.gov/iad/mig/tests/tdt/resources.html (Last Update: 2008)

Jianshu Weng, Erwin Leonardi, Francis Lee. Event Detection in Twitter. 2011. In Proceeding of ICWSM. AAAI Press.

Xintian Yang, Amol Ghoting, Yiye Ruan, and Srinivasan Parthasarathy. 2012. A framework for summarizing and analysing twitter feeds. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '12). ACM, New York, NY, USA.