TopicSketch: Real-Time Bursty Topic Detection from Twitter

Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang

Abstract—Twitter has become one of the largest microblogging platforms for users around the world to share anything happening around them with friends and beyond. A bursty topic in Twitter is one that triggers a surge of relevant tweets within a short period of time, which often reflects important events of mass interest. How to leverage Twitter for early detection of bursty topics has therefore become an important research problem with immense practical value. Despite the wealth of research work on topic modelling and analysis in Twitter, it remains a challenge to detect bursty topics in real-time. As existing methods can hardly scale to handle the task with the tweet stream in real-time, we propose in this paper TopicSketch, a sketch-based topic model together with a set of techniques to achieve real-time detection. We evaluate our solution on a tweet stream with over 30 million tweets. Our experiment results show both efficiency and effectiveness of our approach. Especially it is also demonstrated that TopicSketch on a single machine can potentially handle hundreds of millions tweets per day, which is on the same scale of the total number of daily tweets in Twitter, and present bursty events in finer-granularity.

Index Terms—TopicSketch, tweet stream, bursty topic, realtime

1 Introduction

With 320 million active users and 1 billion tweets per month,[1] Twitter has become one of the largest information portals that provides an easy, quick and reliable platform for users to share anything happening around them with friends and other followers. In particular, it has been observed that, in certain life-critical disasters, Twitter is the most important and timely source from which people find out and track the breaking news before any mainstream media picks up on them and rebroadcast the footage. For example, in the March 11, 2011 Japan earthquake and subsequent tsunami, the volume of tweets sent spiked to more than 5,000 per second when people post news about the situation along with uploads of mobile videos they had recorded.[2] We call such events which trigger a surge of a large number of relevant tweets *bursty topics*.

Fig. 1 shows an example of a bursty topic on November 1st, 2011. A 14-year-old girl from Singapore named Adelyn (not her real name) caused a massive uproar online after she was unhappy with her mother's incessant nagging and resorted to physical abuse by slapping her mother twice, and boasted about her actions on Facebook with vulgarities. Within hours, it soon went viral on the Internet, trending worldwide on Twitter and was one of the top Twitter trends in Singapore. For many bursty events like this, users would like to be alerted as early as it starts to grow viral. However, it was only after almost a whole day that the first news media report on the incident came out. In general, the sheer scale of Twitter has made it impossible for traditional news media, or any other manual effort, to capture most of such bursty topics in real-time even though their reporting crew can pick up a subset of the trending ones. This gap raises a question of immense practical value: Can we leverage Twitter for automated bursty topic detection in real-time?

Unfortunately, this real-time task has not been addressed by the existing work on Twitter topic analysis. First, Twitter's own trending topic list does not help much as it reports mostly those all-time popular topics, instead of the bursty ones that are of our interest in this work. Second, most prior research works define a bursty topic as a set which consists of few bursty words [8], [17], [25], [28], [29], [32]. As only bursty words are captured, the represented bursty topic is far from informative to reflect what the topic really is. Third, most topic modelling based works study the topics in Twitter in a retrospective off-line manner, e.g., performing topic modelling, analysis and tracking for all tweets generated in a certain time period [11], [30], [31], [35]. While these findings have offered interesting insights into the topics, it is our belief that the greatest value of Twitter bursty topic detection has yet to be brought out, which is to detect the bursty topics just in time as they are taking place. This real-time task is challenging for existing algorithms because of the high computational complexity inherent in the topic models as well as the ways in which the topics are usually learnt, e.g., Gibbs Sampling [14] or variational inference [5]. The key research challenge is how to solve the following two problems in *real-time*: (I) How to efficiently maintain proper statistics to trigger detection; and (II) How to model bursty topics without the chance to examine the entire set of relevant tweets as in traditional topic modeling. While some work such as [28] indeed detects events in real-time, it requires pre-defined keywords for the topics.

1. https://about.twitter.com/company
2. http://blog.twitter.com/2011/06/global-pulse.html

• *W. Xie, F. Zhu, J. Jiang, and E. Lim are with the Living Analytics Research Centre, Singapore Management University, Singapore 178902. E-mail: {wei.xie.2012, fdzhu, jingjiang, eplim}@smu.edu.sg.*
• *K. Wang is with Simon Fraser University, BC V5A 1S6, Canada. E-mail: wangk@cs.sfu.ca.*

*Manuscript received 4 Aug. 2015; revised 7 Apr. 2016; accepted 9 Apr. 2016. Date of publication 20 Apr. 2016; date of current version 5 July 2016. Recommended for acceptance by J. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2556661*

1041-4347 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

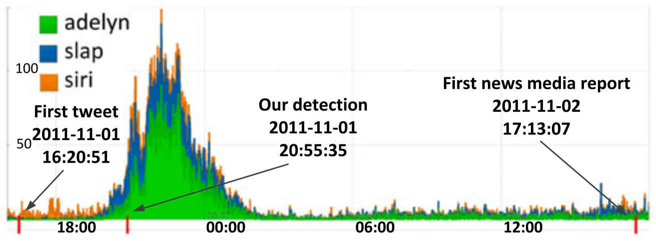


Fig. 1. The tweet volume of each of the top three keywords of the topic: "adelyn", "slap", and "siri".

We propose a new detection framework called TopicSketch. It can be observed from Fig. 1 that TopicSketch is able to detect this bursty topic soon after the very first tweet about this incident was generated, just when it started to grow viral and much earlier than the first news media report.

We summarize our contributions as follows.

First, we proposed a two-stage integrated solution TopicSketch. In the first stage, we proposed a small data sketch which efficiently maintains at a low computational cost the acceleration of two quantities: the occurrence of each word pair and the occurrence of each word triple. These accelerations provide as early as possible the indicators of a potential surge of tweet popularity. They are also designed such that the bursty topic inference would be triggered and achieved based on them. The fact that we can update these statistics efficiently and invoke the more computationally expensive topic inference part only when necessary at a later stage makes it possible to achieve real-time detection in a data stream of Twitter scale. In the second stage, we proposed a sketch-based topic model to infer both the bursty topics and their acceleration based on the statistics maintained in the data sketch.

Second, we proposed dimension reduction techniques based on hashing to achieve scalability and, at the same time, maintain topic quality with robustness.

Finally, we evaluated TopicSketch on a tweet stream containing over 30 million tweets and demonstrated both the effectiveness and efficiency of our approach. It has been shown that TopicSketch on a single machine is able to potentially handle over 150 million tweets per day which is on the same scale of the total number of tweets generated daily in Twitter. We also presented case studies on interesting bursty topic examples which illustrate some desirable features of our approach, e.g., finer-granularity event description.

This work follows the framework of our previous work [33]. While in this work, we provide (I) more sophisticated sketch structure, which captures not only the information of word pairs as in [33], but also the word triples; (II) more effective inference algorithm, i.e., tensor decomposition, which is an important contribution on top of [33]; and (III) more comprehensive evaluations.

2 Related Work

Event detection has been studied for decades, with evolving interests on news [2], [34], blogs [27] and recently social media [25], [28]. As there are numerous research works focusing on it, here we categorised the ones most related to ours, i.e., bursty topic detection in Twitter, along two dimensions: the way of defining a topic and the way of processing data (as shown in Table 1).

TABLE 1
The Related Work is Categorized Along Two Dimensions: The Way of Defining a Topic and the Way of Processing Data

	Clustering Based	Topic Modelling Based
Retrospective	[34]	[35][30][11][31]
Online / Real-time	[34][2][7][26] [28][29][3][24] [25][8][32][17][13]	our work

Clustering based versus topic modelling based. There are different ways to define the topic of an event. In the early work First Story Detection [2], [34] and their successors [7], [26], a topic is represented as a cluster of related documents. By exploiting the temporal proximity of news stories discussing a given event, Yang et al. [34] use refined hierarchical and online document clustering algorithms to detect events from a news stream. In [2] each document is represented as a point in a vector space (e.g., TF-IDF vector), and for each new incoming document, compare it against earlier points. If the new point is close enough to its nearest neighbour, it is absorbed by its nearest neighbour. Otherwise, this new document is labelled as a new event. Brants et al. [7] extend [2] by using incremental TF-IDF model, sophisticated similarity score normalisation etc. However, this approach does not scale to the overwhelming data volume like that of Twitter, as a nearest neighbour search is costly on large data set. Petrovic et al. [26] use locality sensitive hashing (LSH) [20] to scale this approach for Twitter streams. While in other works, a topic is defined as a coherent set (or cluster) of key words [8], [17], [25], [28], [29], [32], hashtags [3], [13], phrases [23] or segments [24]. In such works, usually a collection of bursty terms are detected from the document stream based on some criteria, and possibly later these bursty terms are grouped into several clusters which represent the bursty topics. For instance, the state-of-the-art solution SigniTrend [29] first detects the significant trending terms (words and word cooccurrences) based on its proposed statistic which measures the significances of the terms. With little memory, this statistic can be efficiently updated in an incremental way. What's more, by using hashing techniques, it makes it possible to track all the keyword pairs under a fixed amount of memory. Finally, an end-of-day analysis is performed, which aggregates the detected keywords into larger topics by using clustering approaches.

On the other hand, using a probability distribution over words to represent a topic has been quite common in topic modelling [5], [18]. Especially, the words with high probabilities would characterise the topic well. It is straightforward to learn the topics in the document stream using topic models, and then find the bursty ones by considering their temporal information, such as [11], [30]. Takahashi et al. [30] first use DTM (dynamic topic model) [6] to learn the topics from news stream, and then applies Kleinberg's model [22] to detect the bursty topics. Similarly, Diao et al. [11] build a topic model which simultaneously considers both the temporal information of tweet and user's personal interests to learn the topics from tweet stream, and then Kleinberg's model is used to find the bursty ones. Other

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

related work includes [31] which builds a topic model to find correlated bursty topics from coordinated text streams, and [35] which creates a unified model to distinguish temporal topics from stable topics. In recent works [19], [36] topic models are built to discover geographical topics from Twitter. Although the direct focuses of these works are not on bursty topics, using the similar way as above, geographical bursty topics could be found.

Besides, Ahmed et al. [1] propose a time-dependent topic-cluster model, which combines LDA [5] and clustering to learn the topic of each storyline, and at the same time, to cluster documents into storylines. And similar to [1], in latest work [12] Du et al. cluster continuous-time document streams by proposed Dirichlet-Hawkes Process. Although their model are built on general document streams, they can be potentially adopted to detect bursty topics in Twitter.

Retrospective versus online/real-time. In early work [34] Yang et al. propose methods for both retrospective and online event detection. In the former case, it is assumed that there is a retrospective view of the data in its entirety. On the other hand, in the case of online event detection, the system processes current document before looking at any subsequent documents. It is not surprising that [34] shows the results of retrospective detection are much better than the online one, as more information is available from a retrospective way. As we summarised in Table 1, most topic modelling based methods [11], [30], [31], [35] fall into this category. The complexity of their models makes them good at learning topics from the data in a retrospective way, but at the same time lose the flexibility to respond to any new incoming data. In contrast, methods like [29] only need to maintain a statistic for each term, and report the terms when their statistics exceed the significance level. Under such a framework, online detection is a natural choice. However, the detected topic which consists of few keywords is far less informative than the topic learned from topic modelling, which is a distribution over all the words.

Besides, real-time detection is quite similar to online detection. The subtle difference between them is that in real-time detection, time is crucial, so much so that no fixed time window for detection should be assumed. The only works we are aware of that achieve real-time detection are [28] and [29]. While [28] does detect events in real-time, it needs pre-defined keywords for the topic, making it inapplicable to general bursty topic detection where no prior knowledge of the topic keywords is available. By incrementally updating the statistic in an efficient way, SigniTrend [29] can detect bursty keywords in real-time, but before it aggregates keywords into larger topics, it needs to wait until the end-of-day (or a fixed time period).

In this work, we aim to achieve real-time bursty topic detection from the perspective of topic modelling, which distinguishes us from most existing works in taxonomy (as shown in Table 1). Considering the learning power of topic modelling, we expect our method provide more informative bursty topics than other existing online detection solutions.

3 Solution Overview

3.1 Problem Formulation

A topic in this work is represented as a distribution over words. Particularly, in defining a bursty topic, we evaluate

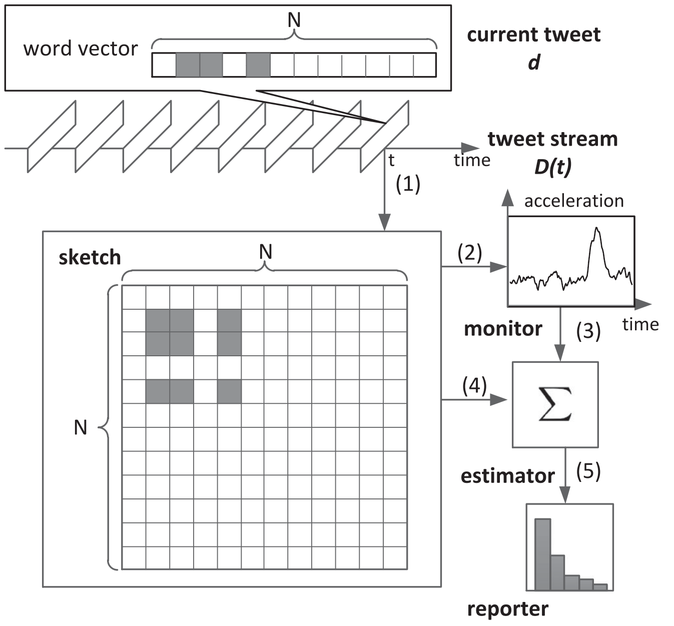


Fig. 2. TopicSketch framework overview.

the following two criteria: (I) There has to be a sudden surge in the topic's popularity which is measured by the total number of relevant tweets. Those all-time popular topics therefore would not count; (II) The topic must be reasonably popular. This would filter away the large number of trivial topics which, despite the spikes in their popularity, are considered as noises because of the negligible number of relevant tweets.

For criterion (I), we measure how bursty a topic is by the *acceleration* of its popularity. Mathematically speaking, acceleration captures the change in the rate of the popularity of a topic. The more sudden the change is, the larger the acceleration is. In Section 4.2, we explain how to estimate the acceleration of a topic without even knowing which tweets are associated to it. For criterion (II), once we found bursty topic candidates, we simply count the relevant tweets of them, and filter out the trivial ones.

Our task in this paper is, given a tweet stream, to detect bursty topics from it as early as possible.

3.2 Solution Overview

Our solution, called TopicSketch, is based on two main techniques—a sketch-based topic model and a hashing-based dimension reduction technique. Our sketch-based topic model provides an integrated two-step solution. In the first step, it maintains as a sketch of the data the acceleration of two quantities: (1) every pair of words, and (2) every triple of words, which are early indicators of popularity surge and can be updated efficiently at a low cost, making early detection possible. In the second step, based on the data sketch, it learns the bursty topics by tensor decomposition. To perform the detection efficiently in large-scale real-time setting, we propose a dimension reduction technique based on hashing which provides a scalable solution to the original problem without compromising much the quality of the topics.

Fig. 2 gives the overview of our proposed TopicSketch framework. The real-time detection flow is as follows: (1) Upon the arrival of each tweet, the sketch is updated, which

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

is an efficient step as detailed in Section 5.2, (2) Once the sketch is updated, the change is sent to the monitor, (3) The monitor tracks the data sketch, compares it with historical average, and triggers the estimator for potential bursty topic detection if the difference is larger than pre-determined threshold. (4) Upon notification, the estimator takes a snapshot of the sketch and infers the bursty topics as described in Sections 4.3 and 5.3, and (5) The inferred bursty topics are sent to the reporter to evaluate and report. TopicSketch is designed such that steps (1) to (3) are computationally cheap to enable real-time response and early detection. Step (4), which is expensive if done naively, is greatly expedited by dimension reduction techniques as described in Section 5.1.

4 Sketch-Based Topic Model

4.1 Intuition

Actually, as mentioned in [17], the term "bursty topic" is very ambiguous, and can be viewed in very different ways. Various intuitions and corresponding definitions on it lead to diverse solutions [3], [17], [22], [29]. The intuition behind this work comes from the observation that, the whole tweet stream is full of large amount of tweets about general topics such as car, music and food. Although they take a large proportion in the whole tweet stream, they are not helpful for our bursty topic detection task. Therefore, we try to separate the bursty topics from them. We found that, following daily routine, people usually tweet about general topics in a steady pace. In contrast, bursty topics are often triggered by some events such as some breaking news or a compelling basketball game, which get a lot of attention from people, and "force" people to tweet about them intensely. In physics, this "force" can be expressed by *"acceleration"*, which in our setting describes the change of *"velocity"*, i.e., arriving rate of tweets. Bursty topics can get significant acceleration when they are bursting, while the general topics usually get nearly zero acceleration. So the *"acceleration"* trick can be used to preserve the information of bursty topics but filter out the others. However, as the topics are hidden, we can not calculate their accelerations directly. A possible way is to estimate them by calculating the accelerations of words instead. Equation (1) shows how we calculate the *"velocity"* $\hat{v}(t)$ and *"acceleration"* $\hat{a}(t)$ of words.

$$\begin{aligned}\hat{v}_{\Delta T}(t) &= \sum_{t_i \le t} X_i \cdot \frac{exp((t_i - t)/\Delta T)}{\Delta T},\\ \hat{a}(t) &= \frac{\hat{v}_{\Delta T_2}(t) - \hat{v}_{\Delta T_1}(t)}{\Delta T_1 - \Delta T_2}.\end{aligned} \tag{1}$$

In Equation (1), X_i is the frequency of a word (or a pair of words, or a triple of words) in the *i-th* tweet, t_i is its timestamp. The exponential part in $\hat{v}_{\Delta T}(t)$ works like a soft moving window, which gives the recent terms high weight, but gives low weight to the ones far away, and the smoothing parameter ΔT is the window size. To capture the change of velocity, acceleration $\hat{a}(t)$ is defined as the difference of velocities with different window size ΔT_1 and ΔT_2. (Similar to the divergence of 5 day average and 10 day average in stock market, which is used to estimate the stock trend.)

We use real data to demonstrate the above intuition in Figs. 3a1, 3b1 and 3c1, respectively present the daily

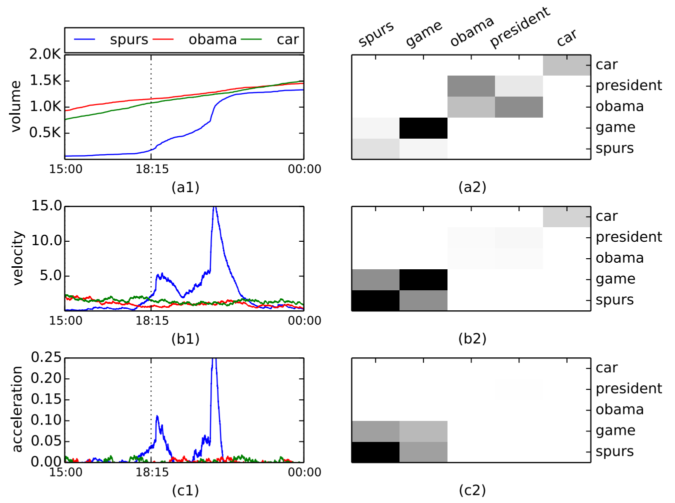


Fig. 3. (a1), (b1), and (c1) show the daily volume, velocity, and acceleration of different keywords over time, respectively; (a2), (b2), and (c2) show the heat maps of these statistics on word pairs at time point 18:15.

volume, velocity and acceleration of three keywords over time, each of which represents a topic. At that day, there was a compelling basketball game between San Antonio Spurs and Oklahoma City Thunder. At the beginning, this event got a big surge in Twitter, and at the end it got another even bigger wave of discussion on this Western Conference final. The daily volume in (a1) shows the popularity of each topic. We can see that, till at the end of the day, "spurs" reaches the same scale as "obama" and "car". However, it will be too late if we wait till we observe the surge in volume to report this bursty topic. As shown in (b1), an earlier indicator is velocity, i.e., the arriving rate of a topic. Our idea of early detection is to monitor the acceleration of a topic which, compared against volume and velocity, gives an even earlier indicator of the popularity surge. The dash line in the plot shows the time when our detection system could be triggered. It is clear that at this time point, the daily volume in (a1) and the velocity in (b1) of "spurs" are not significant enough for identifying this event. In contrast, as "obama" and "car" nearly get zero acceleration in (c1), it is easy to distinguish "spurs" from them under the measurement of acceleration. If we maintain these statistics (i.e., volume, velocity and acceleration) on word pairs instead of single words (use the same formula in Equation (1)), we will get three dynamic matrices over time. (a2), (b2) and (c2) show the heat maps of these matrices at the detection time point respectively. In (a2), the information of all the topics are kept; in (b2) more information are kept for higher velocity topics; and in (c2) only the information of bursty topics are kept, while others are filtered out.

4.2 Sketch-Based Topic Model

Denote $D = \{d_i\}$ as the set of all tweets generated in the tweet stream, where d_i is a tweet in the stream D, and t_i is its timestamp. Also denote $C_{i,w}$ as the number of appearance of word w in tweet d_i, and C_i as the total number of words in tweet d_i. $w \in [N]$, where N is the number of distinct words in the vocabulary.

We consider the following single topic model (shown in Fig. 4). There are K topics $\{\phi_k\}_{k=1}^{K}$, where each topic ϕ_k is a distribution over words. Here the key assumption is each

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

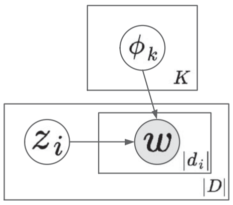


Fig. 4. Single topic model.

tweet d_i is only associated to one latent topic z_i. (This assumption is based on the fact that the length of each tweet is very short, limited by 140 characters.) And each word in d_i is drawn from $Multinomial(\phi_{z_i})$.

Assume topics $\{\phi_k\}_{k=1}^K$ and topic indicator z_i are unknown but fixed, for the frequency of a word w in d_i, denoted as $f_{1,i}[w] = \frac{C_{i,w}}{C_i}$, we have

$$\mathbb{E}[f_{1,i}[w]] = \phi_{z_i}[w].$$

It means the single word frequency $f_{1,i}$ reflects the topic ϕ_{z_i}. However, as the topic indicator z_i is unknown, we can not directly infer topic from it. As shown in Fig. 3, word pairs are useful to group different words into topics, such as the "spurs"-"game" block in the heat maps. We consider calculating the frequency of each word pair. Particularly, with the exchangeability of the words in tweet d_i (based on the single topic model above), we define the frequency of a word pair (w_1, w_2) in a tweet d_i as follows,

$$f_{2,i}[w_1, w_2] = \begin{cases} \frac{P(C_{i,w_1},2)}{P(C_i,2)}, & w_1 = w_2 \\ \frac{C_{i,w_1}C_{i,w_2}}{P(C_i,2)}, & w_1 \neq w_2, \end{cases} \tag{2}$$

where $P(C, 2)$ counts the two-permutations of C. The denominator is the number of all possible cases choosing two out of C_i words in tweet d_i, while the numerator is the number of cases choosing two specific words. Notice that $\sum_{w_2 \in [N]} f_{2,i}[w_1, w_2] = f_{1,i}[w_1]$, which means $f_{2,i}$ actually keeps all the information in $f_{1,i}$. As the words in d_i are drawn from the same distribution $Multinomial(\phi_{z_i})$, it can be proven that

$$\mathbb{E}[f_{2,i}[w_1, w_2]] = \phi_{z_i}[w_1] \cdot \phi_{z_i}[w_2].$$

For simplicity's sake, we adopt the notation of tensor product $\otimes$[3], and denote $f_{2,i}$ as a matrix in which the $[w_1, w_2]$ element is $f_{2,i}[w_1, w_2]$. So we have an equivalent equation.

$$\mathbb{E}[f_{2,i}] = \phi_{z_i} \otimes \phi_{z_i}.$$

As shown in the intuition part 4.1, we calculate the acceleration of word pairs to preserve most of the information for bursty topics, and at the same time filter other general topics out. Set $X_i = f_{2,i}[w_1, w_2]$ and apply Equation (1), we can calculate the acceleration of any word pair (w_1, w_2). Notice that the acceleration $\hat{a}(t)$ in Equation (1) is indeed a weighted sum of sequence $\{X_i\}$, and the weight only depends on $\{t_i\}$. In other words, $\hat{a}(t)$ can be defined as a linear function $\mathcal{A}_t(\cdot)$ on $\{X_i\}$. *Explicitly we denote $\mathcal{A}_t(\{X_i\})$ as the acceleration $\hat{a}(t)$ on $\{X_i\}$.*

For each tweet d_i, we calculate the word pair frequency matrix $f_{2,i}$, so that we have a sequence of matrices, i.e., $\{f_{2,i}\}$. Based on $\{f_{2,i}\}$, we calculate the acceleration of each word pair frequency, i.e., $\mathcal{A}_t(\{f_{2,i}[w_1, w_2]\})$. So that we have a matrix $\mathcal{A}_t(\{f_{2,i}\})$ in which the $[w_1, w_2]$ element is $\mathcal{A}_t(\{f_{2,i}[w_1, w_2]\})$. Fig. 3c2 illustrates what the matrix $\mathcal{A}_t(\{f_{2,i}\})$ looks like. From the linearity of expectation, we have

$$\begin{aligned} \mathbb{E}[\mathcal{A}_t(\{f_{2,i}\})] &= \mathcal{A}_t(\{\mathbb{E}[f_{2,i}]\}), \\ &= \mathcal{A}_t(\{\phi_{z_i} \otimes \phi_{z_i}\}), \\ &= \mathcal{A}_t(\bigcup_k \{\mathbf{1}_k(z_i) \cdot \phi_k \otimes \phi_k\}), \\ &= \sum_k \mathcal{A}_t(\{\mathbf{1}_k(z_i)\}) \cdot \phi_k \otimes \phi_k. \end{aligned} \tag{3}$$

where $\mathbf{1}_k(\cdot)$ is the indicator function such that if $z_i = k$, $\mathbf{1}_k(z_i) = 1$, otherwise, $\mathbf{1}_k(z_i) = 0$. On the left side of Equation (3), $\mathcal{A}_t(\{f_{2,i}\})$ is observable; while on the right side, $\mathcal{A}_t(\{\mathbf{1}_k(z_i)\})$ is in fact the acceleration of tweets about topic k. Thus Equation (3) gives us a way to link the observable acceleration of word pair frequency to the unknown acceleration of topic k without even knowing which tweets are associated to it. More importantly, Equation (3) actually implies that by maintaining the acceleration of word pair frequency, i.e., $\mathcal{A}_t(\{f_{2,i}\})$, we can preserve the information of the topics with high accelerations (i.e., bursty topics), and at the same time filter out the topics with nearly zero acceleration (i.e., stable general topics). That is exactly what we want.

If for any two topics $k_1 \neq k_2$, ϕ_{k_1} and ϕ_{k_2} are orthogonal, i.e., $\langle \phi_{k_1}, \phi_{k_2} \rangle = 0$, all the topics $\{\phi_k\}$ will be the eigenvectors of $\mathcal{A}_t(\{f_{2,i}\})$. If so, we can just perform one single SVD (Singular Value Decomposition) on $\mathcal{A}_t(\{f_{2,i}\})$ to infer these topics. However, $\langle \phi_{k_1}, \phi_{k_2} \rangle = 0$ means topics k_1 and k_2 have no any common words, which is far from reality. Thus we consider even higher order information, i.e., the frequency of each word triple. Similar to $f_{2,i}$, the frequency of a word triple (w_1, w_2, w_3) in a tweet d_i is defined as follows:

$$f_{3,i}[w_1, w_2, w_3] = \begin{cases} \frac{P(C_{i,w_1},3)}{P(C_i,3)}, w_1 = w_2 = w_3 \\ \frac{P(C_{i,w_1},2)C_{i,w_3}}{P(C_i,3)}, w_1 = w_2 \neq w_3 \\ \frac{P(C_{i,w_2},2)C_{i,w_1}}{P(C_i,3)}, w_2 = w_3 \neq w_1 \\ \frac{P(C_{i,w_3},2)C_{i,w_2}}{P(C_i,3)}, w_3 = w_1 \neq w_2 \\ \frac{C_{i,w_1}C_{i,w_2}C_{i,w_3}}{P(C_i,3)}, otherwise. \end{cases} \tag{4}$$

Similarly, it can be proven that

$$\mathbb{E}[f_{3,i}] = \phi_{z_i} \otimes \phi_{z_i} \otimes \phi_{z_i},$$

and

$$\mathbb{E}[\mathcal{A}_t(\{f_{3,i}\})] = \sum_k \mathcal{A}_t(\{\mathbf{1}_k(z_i)\}) \cdot \phi_k \otimes \phi_k \otimes \phi_k. \tag{5}$$

Based on Equations (3) and (5), we transform our problem into a standard tensor decomposition problem [4], i.e., given sketches $M_2 = \mathcal{A}_t(\{f_{2,i}\})$ and $M_3 = \mathcal{A}_t(\{f_{3,i}\})$, infer

3. For vectors v_1, v_2, $v_3 \in \mathbb{R}^N$, the tensor product of v_1 and v_2, $m = v_1 \otimes v_2$, is defined as a $N \times N$ matrix, where $m[i,j] = v_1[i]v_2[j]$, $i, j \in [N]$. And the tensor product of v_1, v_2 and v_3, $m = v_1 \otimes v_2 \otimes v_3$, is defined as a $N \times N \times N$ matrix, where $m[i,j,k] = v_1[i]v_2[j]v_3[k]$, $i, j, k \in [N]$.

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

topics $\{\phi_k\}$. In Section 4.3 we present the tensor decomposition algorithm in detail.

So by now, we have two sketches: M_2 and M_3. And each cell in these two matrices is an acceleration. For instance, the (w_1, w_2) cell in M_2, i.e., $M_2[w_1, w_2]$, is $\mathcal{A}_t(\{f_{2,i}[w_1, w_2]\})$. Notice that all these accelerations are easy to compute and update upon the arrival of every tweet (as shown in Section 5), which is critical for scalability in real-time setting.

As M_3 could be huge (a $N \times N \times N$ matrix), we do not really store the M_3, but project it to a $N \times N$ matrix $M_3(\eta) = \sum_w M_3[:, :, w] \cdot \eta[w]$, where $\eta \in \mathbb{R}^N$, is a random vector. However a $N \times N$ matrix is still huge, in Section 5, we discuss more about reducing the space complexity.

4.3 Topic from Sketch

To identify bursty topics from the data sketch, which consists of two matrices M_2 and $M_3(\eta)$, we employ a tensor decomposition algorithm in [4]. This algorithm first performs a SVD on M_2 to find a whitening matrix W. Afterwards, whiten $M_3(\eta)$, then perform another SVD on whitened $M_3(\eta)$ to find its eigenvectors, from which the topic vectors can be recovered. The detail is presented in Appendix A. Here we give the procedure in Algorithm 1. It has three parts: (I) Whitening, which transform $M_3(\eta)$ from a $N \times N$ matrix to a $K \times K$ matrix T_3; (II) SVD, which gets the generalized vectors $\{v_k\}$ of T_3; (III) Reconstruction, which recovers the topics $\{\phi_k\}$ and their corresponding accelerations $\{a_k\}$. As $K \ll N$, the time consuming part here is part (I), which takes time in the order of $O(K \cdot N^2)$.

Algorithm 1. TensorDecompose

```
Input: $K$: the number of topics.
       $M_2$: the second order tensor power.
       $M_3(\eta)$: the reduced third order tensor power.
Output: topics $\{\phi_k\}$ and their corresponding accelerations
        $\{a_k\}$.
1  /* Whitening */
2  /* $eigs(M_2, K)$ returns the largest $K$ largest magnitude
   eigenvalues and corresponding eigenvectors. */
3  $(U, \Lambda) = eigs(M_2, K)$;
4  $W = U\Lambda^{-\frac{1}{2}}$; /* $W$ may be a complex matrix */
5  $T_3 = W^\top M_3(\eta) W$
6  /* SVD */
7  Compute generalised vectors $\{v_k\}$ of $T_3$;
8  /* Reconstruction */
9  for $k = 1$ to $K$ do
10     $\phi_k = \frac{W(W^\top W)^{-1} v_k}{1_N^\top W(W^\top W)^{-1} v_k}$;
11     $a_k = \frac{1}{(W^\top \phi_k)^\top (W^\top \phi_k)}$;
12 end
13 return$\{\phi_k\}, \{a_k\}$.
```

5 Realtime Detection Techniques

In this section, we present the technique details to achieve real-time efficiency for bursty topic detection in the huge-volume tweet stream setting.

5.1 Dimension Reduction

The first challenge is the high dimension problem as a result of the huge number of distinct words N in the tweet stream, which could easily reach the order of millions or even larger (see the experiments in Section 6.1). What's more, user-generated new words or hashtags always appear in Twitter. This results not only in an enormous data sketch (recall M_2 and $M_3(\eta)$ in the sketch are $N \times N$ matrices) but also a very high dimension input to Algorithm 1.

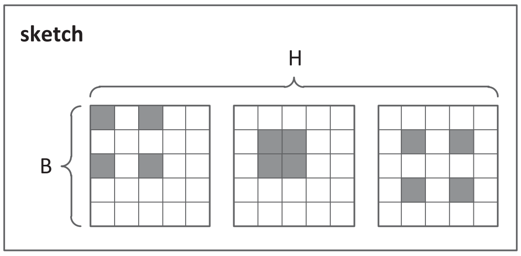


Fig. 5. New sketch after dimension reduction.

Since the problem is mainly because N is too large, one natural solution is to keep only a set of active words encountered recently, e.g., in the last 15 minutes. When a burst is triggered, consider only the words in this recent set. However, it turns out that the size of this reduced active word set for tweet stream is still too large (see Section 6.1) to infer the topics efficiently.

To handle large number of words, another common way is hashing [29]. We hash these distinct words into B buckets, where B is a number much smaller than N, and treating all the words in a bucket as one "word". Consequently, the size of the sketch becomes $O(B^2)$, which are significantly smaller than $O(N^2)$ as in the original problem. However, after hashing, what we obtain is the distribution over buckets, rather than the distribution over words. It means we would need to recover the probabilities of words from the probabilities of buckets. To solve this problem, we adapt the Count-Min algorithm [10], [21] to our setting, by using H hashing functions instead of one. In particular, it works as follows. Given H hash functions $(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_H)$ which map words to buckets $[1...B]$ uniformly and independently. For a topic k with word distribution ϕ_k, we first estimate its distribution over buckets $\{\phi_k^{(h)}[j] = \sum_{i|\mathcal{H}_h(i)=j} \phi_k[i]\}_{j=1}^B$ for all the hash functions. Then we recover the probability of each word i as $\min_{1 \le h \le H}\{\phi_k^{(h)}[\mathcal{H}_h(i)]\}$.

The sketch after hashing is illustrated in Fig. 5. As shown in Fig. 5, for each arriving tweet, H different bucket-pair frequencies are calculated, and H matrices are updated correspondingly. After the dimension reduction, the memory cost for the sketch is $O(H \cdot B^2)$, and the time complexity for tensor decomposition is $O(H \cdot B^2 \cdot K)$, which are small enough to be practically feasible.

We also maintain a pool of active words, so that we estimate the probability of words only in this set rather than all the words in whole vocabulary. The procedure is presented in Algorithm 2. This algorithm will estimate the probability of each word with error no greater than $\frac{e}{B}$ with a probability of e^{-N/e^H}. The details of the proof can be found in [21].

5.2 Efficient Sketch Maintenance

The core part of sketch maintenance in our solution is acceleration calculation. As presented in Section 4.1, we adopt Equation (1) to calculate acceleration. However, directly applying Equation (1) is far from efficient. Observed that

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

the velocity $\hat{v}(t)$ in Equation (1) can be calculated in an incremental way as in the following Equation (6),

$$\hat{v}_{\Delta T}(t) = \begin{cases} \hat{v}_{\Delta T}(t_{i-1}) \cdot e^{\frac{(t_{i-1}-t)}{\Delta T}}, & t \in (t_{i-1}, t_i) \\ \hat{v}_{\Delta T}(t_{i-1}) \cdot e^{\frac{(t_{i-1}-t)}{\Delta T}} + \frac{X_i}{\Delta T}, & t = t_i. \end{cases} \tag{6}$$

where t_i is the timestamp of the ith tweet. So instead of directly storing one acceleration, we incrementally maintain two velocities $\hat{v}_{\Delta T_1}(t)$ and $\hat{v}_{\Delta T_2}(t)$, from which the acceleration can be derived on the fly. Both the space complexity for maintaining one acceleration and the time complexity for updating one acceleration are therefore $O(1)$.

Algorithm 2. TopicRecover

```
Data: $active\_words$ : the pool of active words.
      $\{\mathcal{H}_h\}_{h=1}^{H}$ : $H$ hash functions.
      $threshold$ : pre-defined threshold.
Input: $\{\phi_k^{(h)}\}_{h=1}^{H}$ : $H$ distributions over $[B]$.
Output: $topic_k$ : a topic which is represented as a set of words.
1 initialise $topic_k$ as a empty set.;
2 for each word w in active_words do
3     if $\min_{1\le h\le H}\{\phi_k^{(h)}[\mathcal{H}_h(w)]\} \ge threshold$ then
4         add $word$ to $topic_k$;
5     end
6 end
7 return $topic_k$
```

Another issue is that, there are $O(N^2)$ accelerations to be updated when each tweet arrives, and $O(H \cdot B^2)$ for the dimension reduced case. We take lazy maintenance strategy to reduce the computing cost. In fact, for each acceleration, take the (w_1, w_2) cell in M_2 as an example, we store a velocity pair $(\hat{v}_{\Delta T_1}(t), \hat{v}_{\Delta T_2}(t))$ and a timestamp $t^\star$ representing the last modification time. When a tweet d_i arrives, if $C_{i,w_1} \cdot C_{i,w_2} > 0$, we update the pair $(\hat{v}_{\Delta T_1}(t), \hat{v}_{\Delta T_2}(t))$ according to Equation (6) and update timestamp $t^\star$ to t_i, otherwise we adopt lazy strategy and simply do nothing. When a bursty topic is triggered at time t, take a snapshot of the sketch (copy on write), then recover all the velocity pairs up to time t by deriving the accelerations from the velocity pairs according to Equation (6). Fig. 2 illustrates this updating procedure: When a tweet arrives, its word vector is shown. The gray cells in the word vector denotes the occurrence of relevant words in the tweet. In this example, three words occur in the current tweet. The gray cells in sketch represents the updated elements in the sketch. $O(|d|^2)$ cells are updated in total. Fig. 5 illustrates this updating procedure for dimension reduced case. After hashing, three words in the current tweet are mapped into two buckets for three hash functions respectively. In total, $O(H \cdot |d|^2)$ cells are updated.

5.3 Topic Inference

Once our system is triggered, we take a snapshot of the sketch, and then infer the bursty topics from the sketch as shown in Section 4.3. As discussed in Section 5.1, instead of directly maintaining a sketch with size of $N \times N$, we maintain H sketches with size of $B \times B$, where B is much smaller than N. Here we show how to infer the bursty topics from the sketch after dimension reduction. First, we get topics from the H different sketches by *TensorDecompose* (Algorithm 1). Note this step can be implemented in parallel. One subtle problem here is that a bursty topic may have different topic indexes in different sketches (recall we have H different sketches now). For instance, sketch $h = 1$ may capture a bursty topic in topic $\phi_1^{(1)}$ with index 1, however sketch $h = 2$ may capture the same bursty topic in topic $\phi_3^{(2)}$ with index 3. To align the topics which represent the same bursty topic, we sort these topics $\{\phi_k^{(h)}\}_{k=1}^{K}$ by their corresponding accelerations $\{a_k^{(h)}\}_{k=1}^{K}$, and then re-index them according to the order. At last, we get the topics recovered by *TopicRecover* (Algorithm 2). An overview of this procedure is given in Algorithm 3. For the purpose of robust, we also propose a variant of this algorithm by using the second minimum value instead of the minimum value in Line 3 in Algorithm 2. We will discuss this more in the experiment part.

Algorithm 3. BurstyTopicsDiscover

```
Data: $active\_words$ : the pool of active words.
      $\{\mathcal{H}_h\}_{h=1}^{H}$ : $H$ hash functions.
      $threshold$ : pre-defined threshold.
Input: $K$: the number of topics
Input: $\{M_2^{(h)}\}_{h=1}^{H}\{M_3(\eta)^{(h)}\}_{h=1}^{H}$: $H$ sketches.
Output: $\{topic_k\}$ : a list of topics.
1  /* get topics from $H$ different sketches */;
2  for h = 1 to H do
3      /* in parallel */;
4      $\{\phi_k^{(h)}\}_{k=1}^{K}, \{a_k^{(h)}\}_{k=1}^{K} = TensorDecompose(K, M_2^{(h)}, M_3(\eta)^{(h)})$
5  end
6  /* aligning topics*/;
7  for h = 1 to H do
8      sort $\{\phi_k^{(h)}\}_{k=1}^{K}$ by their corresponding $\{a_k^{(h)}\}_{k=1}^{K}$;
9  end
10 /* topic discovering */;
11 for k = 1 to K do
12     $topic_k = TopicRecover(\{\phi_k^{(h)}\}_{h=1}^{H})$;
13 end
14 return$\{topic_k\}$
```

After getting the topics from Algorithm 3, we would like to add a heuristic step to refine the topics, due to the following reasons. First, because of the original acceleration based design, our solution is fragile when facing spam accounts. They usually inject a lot of duplicate or similar tweets very intensively in a short period of time, which would produce words with significant acceleration, and therefore trigger our system. Second, due to hashing collision, it is possible that some rare words appear in the recovered topic. The refining process is as follows:

- *Trivial topic filtering*: filter out the topics with small number of relevant tweets.
- *Noisy topic filtering*: filter out the topics with high entropy, in which there are no high probability words. Such topics look like noises.
- *Spam filtering*: check the related tweets of a topic in recent 5 minutes. If there is an account which posts a significant number of tweets, filter out this topic.
- *Rare word pruning*: prune the words which do not appear in the recent tweets.

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

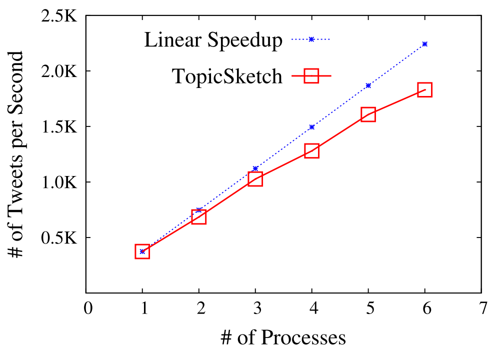


Fig. 6. Comparison between throughputs for TopicSketch and Linear Speedup for varying numbers of processes.

An inverted index of recent tweets is implemented, so that this step can be performed efficiently.

6 Evaluation

In this section, we present the evaluation of our TopicSketch system for both efficiency and effectiveness. We use two different Twitter data sets: one consists of tweets from Singapore, the other consists of tweets from San Francisco. These tweets are crawled from the Twitter users whose profile locations are Singapore and San Francisco, respectively. The Singapore based data set contains 32,479,134 tweets, and the San Francisco based data set contains 99,586,724 tweets. These tweets are used to simulate live tweet streams. We implemented our prototype system in Python 2.7[4] using 64-bit addressing, and executed on multiple cores of an Intel Xeon 3.06 GHz machine. To demonstrate how TopicSketch detects bursty topics on the fly, a demo is presented at `http://topicsketch.appspot.com/` for exploration (best use Chrome browser).

6.1 Efficiency Evaluation

In this section, we evaluate the performance of the sketch maintenance by the throughput on the tweet stream. We also evaluate the performance of the estimator by the topic inference time. In our tweet set, after removing stop words, the average number of words in a tweets is 8 (evidence for small $|d|$). The total number of the distinct words is 8,470,180 (evidence for high dimensions). The number of distinct words in the 15 minutes active word set is between 10,000 and 20,000.

6.1.1 Sketch Maintenance

Before each incoming tweet goes to the component of sketch maintenance, we conduct standard preprocessing on it, including tokenizing,[5] stemming, and stop words removing. After preprocessing, calculate word pair frequency $f_{2,i}$ and word triple frequency $f_{3,i}$ as shown in Equations (2) and (4). Then update the accelerations in sketch according to Equation (6) and our lazy maintenance strategy in Section 5.2.

According to Equations (2) and (4), the time complexity of frequency matrix computation is $O(H \cdot |d|^3)$. And the time complexity for sketch maintenance is $O(H \cdot |d|^2)$ for each incoming tweet, according to the analysis in Section 5.2.

4. For the reason of efficiency, some core part such as sketch maintenance was implemented in C++.

5. The Tweet NLP tool (http://www.ark.cs.cmu.edu/TweetNLP) is used.

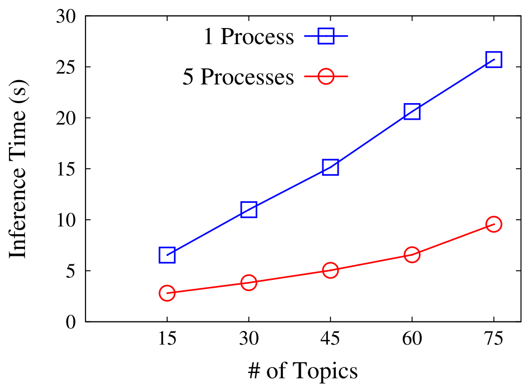


Fig. 7. The inference time for varying numbers of topics.

It is not hard to maintain the sketch in parallel on multiple cores in one machine. Partition the stream D into S parts, i.e., $D = \cup_{s=1}^{S}\{D_s\}$. For each sub-stream D_s, we can maintain a sketch, which consists of two matrices $M_2^{(s)} = \mathcal{A}_t(\{f_{2,i}\}_{d_i \in D_s})$ and $M_3^{(s)} = \mathcal{A}_t(\{f_{3,i}\}_{d_i \in D_s})$. As $\mathcal{A}_t$ is in fact a linear function, the sketch on stream D is equal to the sum of the sketch on each sub-stream D_s, i.e., $M_2 = \mathcal{A}_t(\{f_{2,i}\}) = \mathcal{A}_t(\cup_{s=1}^{S}\{f_{2,i}\}_{d_i \in D_s}) = \sum_{s=1}^{S} \mathcal{A}_t(\{f_{2,i}\}_{d_i \in D_s}) = \sum_{s=1}^{S} M_2^{(s)}$. The same, $M_3 = \sum_{s=1}^{S} M_3^{(s)}$. According to this, we maintain the sketch on multiple cores on a single machine. In particular, we build a process pool of size S, and each process in the pool is in charge of a sub-sketch. Notice that we do not partition the stream in advance. Instead, for each arriving tweet, randomly send it to one process in the pool on the fly.

To evaluate the throughput of the sketch maintenance and its scalability, we set the number of processes in the process pool from 1 to 6, respectively. Fig. 6 shows the throughput for process pool of different sizes. Notice that when the number of the processes is 6, the throughput is 1,830 tweets per second (over one hundred and fifty millions tweets per day), which is on the same scale of the total number of tweets generated daily in the whole Twitter network. We can observe that the throughput linearly increases as the number of processes increases, which shows the scalability of TopicSketch. Also observe that the curve is a little away from the ideal case – linear speedup (the dashed line in Fig. 6). It may be because of the additional cost for communication between processes.

6.1.2 Topic Inference

As described in Section 5.3, we can infer the bursty topics from H different sketches in parallel. In particular, we built a process pool and each process in the pool is in charge of performing *TensorDecompose* for one sketch under hash function $\mathcal{H}_h$. We set the size of process pool to 1 and 5, which represent the sequential version and fully parallel version respectively. We vary the number of topics K from 15 to 75. Fig. 7 shows the performance of the algorithm. The result shows that although the sequential version gives an affordable running time (less than half minute), the parallel version provides significant improvement (10 seconds for 75 topics). Also observe that for both versions, as the

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

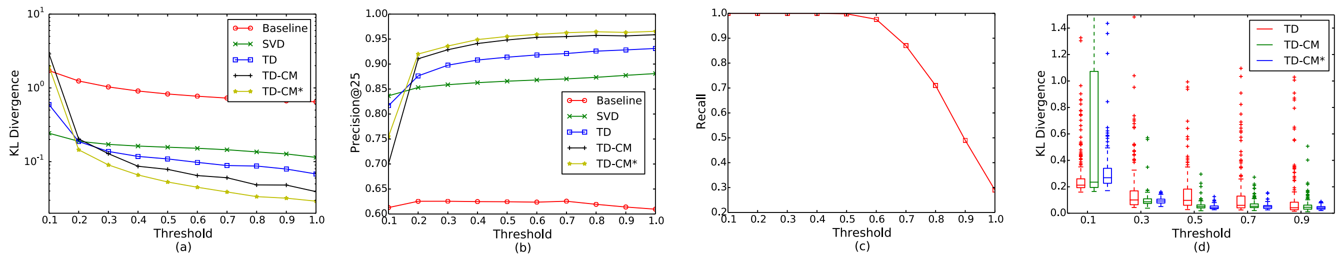


Fig. 8. (a)-(b) Comparison of KL Divergence and Precision at top 25 words for varying thresholds. (c) Recall for varying thresholds. (d) KL Divergence for 200 different random vector η.

number of topics increases, the inference time linearly increases, which supports our analysis about the time complexity of Algorithm 1, and shows the scalability of the inference algorithm.

6.2 Effectiveness Evaluation

6.2.1 Evaluation on Synthetic Data

We first use synthetic data to verify the correctness of our solution. As mentioned in Section 4.2, we model the tweet stream as a mixture of tweets of different topics, and each tweet is only associated to one topic. We assign a temporal point process [15] and a distribution over words to each topic, then sample time stamps based on its point process, and sample tweets according to its distribution. We consider three types of topics: 1) the general topics (e.g., sport, education, car) which have steady arriving rates and take a large proportion in the whole stream; 2) the bursty topics which have highly intensive arriving rates after they go viral; 3) the noisy topics which are the small spikes in the stream. For general topics, the homogeneous Poisson point process is used to simulate the arriving rates of their tweets. And we draw the Poisson parameter λ from log-normal distribution to model different arriving rates for different topics. The Hawkes process [16] is used to simulate the arriving rates for bursty and noisy topics. Specifically, we adopt the following equation to calculate the arriving rate.

$$\lambda(t) = \lambda_0 e^{-(t-t_0)/w} + \alpha \sum_{t_i<t} e^{-(t-t_i)/w},$$

where t_0 is when the topic is generated, λ_0 is the initial intensity parameter and w is the decaying parameter. The second part in above equation mimics the self-enforcing phenomenon in Twitter, and exponential term models the waning of popularity over time.

In the synthetic data stream, we simulate 50 general topics, one bursty topic and five noisy topics. For each general topic, draw $\lambda \sim log\mathcal{N}(1,1)/60$ as its parameter. For bursty topic, the initial intensity λ_0 is set to 2.0. For the five noisy topics, λ_0 is set to 0.1, 0.2, ... , 0.5, respectively. For both bursty topic and noisy topics, $w = 100$ and $\alpha = 0.009$. All the topics are drawn from Dirichlet distribution. Particularly, we choose such noisy topics that the inner product of any noisy topic and the bursty topic is at least 0.01. And the vocabulary size is 5,000.

KL-divergence is used to compare the uncovered bursty topic with the ground-truth bursty topic. The precision at top 25 words in the bursty topic is also calculated. We set different thresholds for detection. All experiments are repeated 500 times.

We compare the following methods.

- *Baseline*: the topic is estimated based on the acceleration vector of single word frequency, i.e., $\mathcal{A}_t(\{f_{1,i}\})$.
- *SVD*: perform a SVD on the acceleration matrix of word pair frequency $\mathcal{A}_t(\{f_{2,i}\})$, and return the primary eigenvector.
- *TD*: perform the tensor decomposition in Algorithm 1, and return the topic with highest acceleration.
- *TD-CM*: tensor decomposition together with the dimension reduction technique Count-Min as shown in Algorithm 3.
- *TD-CM**: a variant of TD-CM, the subtle difference is that TD-CM* uses the second minimum value instead of the minimum value in Line 3 in Algorithm 2 for the purpose of robustness.

Notice that for the evaluation on synthetic data, no additional refining step is added. For TD-CM and TD-CM*, we use $H = 5$ hash functions, set the bucket size $B = 1{,}000$ and pick the number of topics $K = 6$. Figs. 8a and 8b show the average performances for each of above methods. Fig. 8c shows the recall of our acceleration based detection method for varying thresholds. As expected, higher threshold leads to better performance, but lower recall. The same as many other threshold based solutions, we need to find a good trade-off between recall and precision.

To our surprise, we found that, after dimension reduction, TD-CM and TD-CM* even have better performances than the original TD. The reason is that the tensor decomposition method in Algorithm 1 relies on a "good" random vector η, such that for any $k_1 \neq k_2$, $\langle \eta, \phi_{k_1} \rangle \neq \langle \eta, \phi_{k_2} \rangle$ (See Appendix A). In high dimension space, there is higher chance to "violate" this condition than in lower dimension space. To verify this, on one synthetic data stream, we generate 200 random vectors, and see how different choices of η affect the performances. Fig. 8d illustrates that, although in average TD can get good performance, TD has much more outliers than TD-CM and TD-CM*. Considering the risk of failure due to a "bad" choice of η, in TD-CM* we always assume there maybe one copy of sketch in H fails to recover the bursty topic, and modify Count-Min by using the second minimum value instead of the minimum value in Line 3 in Algorithm 2. Fig. 8d shows TD-CM* has the most robust performance.

6.2.2 Evaluation on Real Data

In this section, we examine our solution on two real Twitter data sets: Singapore based data set and San Francisco based data set. Unlike synthetic data, Twitter data set does not come with ground truth, i.e., the bursty topics. For evaluation, one indirect way is to identify bursty word pairs from

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

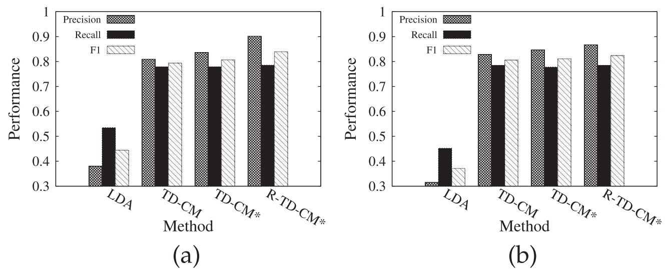


Fig. 9. Performance on (a) San Francisco data and (b) Singapore data.

tweet stream first (in a retrospective way), and then use these bursty word pairs as "ground truth" to measure the performance of our solution. The underlying logic is as follows. For each bursty word pair, we expect our solution can detect some bursty topic which contains this bursty word pair in its top words. On the other hand, for each detected bursty topic, we expect it contains some bursty word pair.

First, we count every word pair by day. Then for each word pair, we find the date when its daily count is far beyond its average. Particularly, we employ the following equation proposed in [29] to calculate the significant score for each word pair.

$$sig_\beta(c) = \frac{c - max\{\mu, \beta\}}{\sigma + \beta},$$

where c is the daily count of a word pair, μ is the average, σ is the standard deviation and β serves as a noise filter. $sig_\beta(c)$ here works like the z-score in the case of normal distribution. For a word pair (w_1, w_2) on some $date$, if its $sig_\beta(c) > 3$, then it is a bursty word pair on that date, and we generate a tuple $[date : (w_1, w_2)]$. After processing all the word pairs, we get a list of tuples. Based on this list, we measure the precision of our solution by the fraction of detected bursty topics that are "supported" by at least one tuple $[date : (w_1, w_2)]$, i.e., the bursty topic should contain (w_1, w_2) in its top 10 words, and it must be detected on $date$. For the recall, we get the top 1,000 bursty word pairs according to their daily counts, and calculate the recall as the fraction of these bursty word pairs that are "reflected" in some detected bursty topic, which contains (w_1, w_2) in its top 10 words, and is detected on $date$.

In this experiment, we conduct the following baseline: once our detection system is triggered, train the LDA model using the recent 15 minutes tweets, and after Gibbs sampling, return the topic with largest number of assigned words. We use 15 days tweets to tune parameters (e.g., threshold and number of topics) for both LDA and our solutions. Fig. 9 shows the performances of different methods on both Singapore based tweets and San Francisco based tweets. TD-CM, TD-CM* are the same as Section 6.2.1. R-TD-CM* is the solution of TD-CM* followed by a refining step presented in Section 5.3. It shows that, overall our solutions outperform LDA. Further more, TD-CM, TD-CM* and R-TD-CM* have roughly the same recall, and because of the refining step, R-TD-CM* gets higher precision than TD-CM and TD-CM*.

The above precision which is measured by bursty word pairs actually cannot fully reflect the quality of detected topics. A topic which contains some bursty word pair may

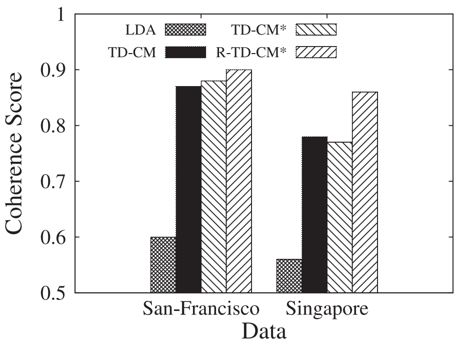


Fig. 10. The coherence scores of the detected bursty topics from different methods.

also have other irrelevant words as its top words. To measure the coherence of the detected bursty topics, we adopt the *word intrusion* task proposed in [9]. In this task, the subject is presented with six randomly ordered words, in which five words are from a detected bursty topic and the other one is a *intruding* word. The task of the user is to identify the *intruding* word which is not related to this bursty topic. Different from the task in [9], if only presented with the words, it is not easy for human user to find out the intruding word. For instance, when presented with {"spurs", "horse", "thunder", "99-107", "game", "fisher"}, without exploring the related tweets, a human user, who is not familiar with USA basketball, may not know this set of words (except the intruder "horse") actually is about a basketball match, consequently, she may identify a wrong intruder. So in our task, the words are presented together with an interactive plot which shows the counts of each word over time, and the tweets contains these words, within a time window around the detection time. (See a demo here.[6]) After exploring all the above information, if the presented set of words lacks coherence, the human user would be still confused, and just pick an intruder at random.

For each detected bursty topic, we first select five words with highest probabilities from it. Then the intruding word is selected at random from a pool of words which are popular in the tweet stream on the detection date. We chose 50 bursty topics randomly for each method. Three human users are asked to perform this task. We calculate the coherence score as $\sum_{i=1}^{50}\sum_{j=1}^{3}\mathbf{1}(intruder_i = word_{ij})/150$, where $intruder_i$ is the intruding word for the ith bursty topic, $word_{ij}$ is the chosen word from human user j for the ith bursty topic. Fig. 10 shows the comparison between different methods in terms of coherence score.

6.2.3 Parameter Analysis

Here we study the effect of bucket size B. Fig. 11 shows the performances of TD-CM* for different bucket sizes on synthetic data. We can observe the improved performance as bucket size B increases. From $B = 500$ to $B = 1{,}000$, the improvement is significant, while from $B = 1{,}000$ to $B = 1{,}500$, the improvement is little. Fig. 12 shows how the bucket size B affects the performance of TD-CM* on real data. As expected, it shows that better performance can be achieved when we set bigger bucket size B. However,

6. http://topicsketch.appspot.com/

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

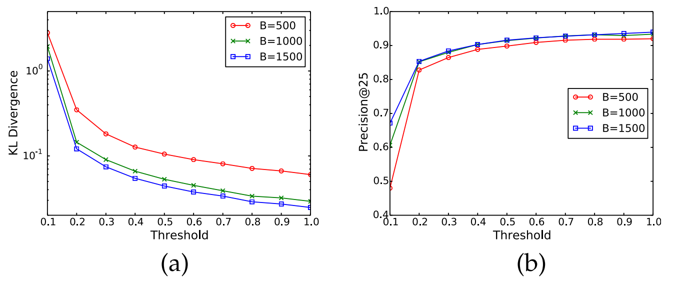


Fig. 11. Varying bucket size B, performances on synthetic data: (a) KL Divergence and (b) Precision at top 25.

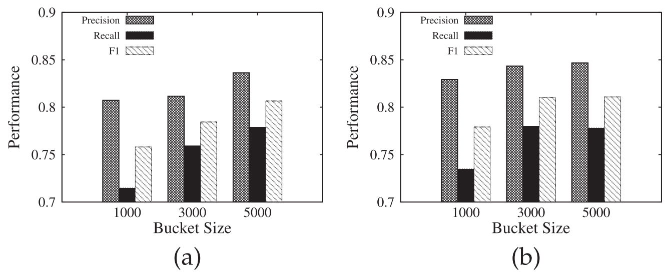


Fig. 12. Varying bucket size B, performances on (a) San Francisco data and (b) Singapore data.

bigger bucket size means higher computational cost. Fig. 7 shows the computational cost is acceptable when $B = 5{,}000$. So in the paper, rather than explore other even bigger bucket size, we empirically set $B = 5{,}000$.

We also check the effect of smoothing parameters ΔT_1 and ΔT_2 on synthetic data. Figs. 13a, 13b show how different settings of ΔT_1 and ΔT_2 affect the performance of TD-CM* in terms of KL divergence and recall. Basically, bigger smoothing parameters leads to higher precision but lower recall. In practice, we test different settings of smoothing parameters, and choose the one which provides a good balance between precision and recall.

6.3 Comparison with Other Solutions

6.3.1 *Comparison with* Twevent

We compare TopicSketch to previous Twitter event detection system Twevent [24] by case studies. We adopt the same dataset as used in the original paper [32]. We present in Table 2 an event detected by both algorithms[7]—Apple WWDC 2010, to show the differences between TopicSketch and Tewent. We manually group together sub-events belonging to a single larger event. Table 2 demonstrates TopicSketch's ability to describe events at a finer granularity. On June 7, 2010, Steve Jobs announced the release of the fourth generation iPhone, causing huge wave in Twitter. At this WWDC, several new features of this new generation iPhone were introduced, including farmville client, retina display and iMovie. As shown in Table 2, TopicSketch not only detected the big event of Apple WWDC 2010 as a whole, it also detected a sequence of sub-events. In particular, each sub-event in the table which triggers our system indeed corresponds to a highlight of the WWDC event, from which we can tell those new features of iPhone that users find more interesting than others.

7. We referred the results listed in Table 2 from [24].

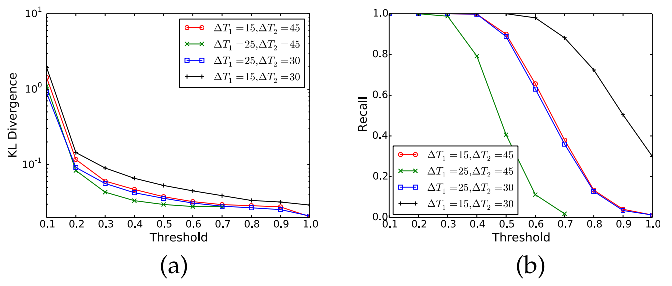


Fig. 13. Varying smoothing parameters ΔT_1 and ΔT_2, performances on synthetic data: (a) KL Divergence and (b) Recall.

6.3.2 *Comparison with* SigniTrend

SigniTrend is the state-of-the-art emerging topics detection solution [29]. SigniTrend performs a two-stage detection: first detects the bursty words (and word-pairs) whose significance score exceed a pre-defined threshold on the fly, and then performs end of day analysis to cluster the bursty words into larger topics. Similarly, TopicSketch also first detects the bursty word-pairs according to their accelerations on the fly, and once the system is triggered immediately infer the bursty topics from the sketch. As the detail of clustering step is not presented in [29], we do not directly compare the results from these two solutions. Instead, we check how different triggers affect the performance of our solution. Particularly, we use the significance score from SigniTrend instead of the acceleration in our paper as the trigger, and the inference part after triggering remains the same. We also use 15 days tweets to tune the parameters. Figs. 14 and 15 show the performances of TD-CM* and R-TD-CM* for different triggers respectively. (SIGNI stands for significance score and ACCEL for acceleration). It can be observed that, roughly, using significance score as the trigger can achieve higher precision while using acceleration as the trigger can achieve higher recall.

6.3.3 *Limitation of TopicSketch*

Compared with Twevent and SigniTrend, TopicSketch is fragile when facing spam accounts. For instance, some spam account may abuse key word "music" by injecting a lot of tweets about some music album into tweet stream in a very short period of time. This would produce words with significant acceleration, and therefore trigger our detection system. After the topic inference step, our system would report a topic about this music album, which is not popular at all. Thus in Section 5.3, a refining step is proposed to remedy this situation.

Besides, we also found that, due to the single topic assumption, TopicSketch is not suitable for stream of documents with multiple topics.

7 Conclusions

In this paper, we proposed TopicSketch a framework for real-time detection of bursty topics from Twitter. Due to the huge volume of tweet stream, existing topic models can hardly scale to data of such sizes for real-time topic modeling tasks. We developed a "sketch of topic", which provides a "snapshot" of the current tweet stream and can be updated efficiently. Once burst detection is triggered, bursty topics can be inferred from the sketch efficiently. Compared with existing event detection system, from a different perspective—the

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
List of Events Detected by TopicSketch and Twevent

Event	Sub-Event	TopicSketch	Twevent
WWDC2010	Farmville client for iPhone 4 was demonstrated.	#wwdc, farmville, iphone, zynga, netflix, god, ipad, wwdc, comes, soon	
	Retina display of iPhone 4 was introduced.	iphone, #wwdc, retina, display, pixels, crystal, clear, 326, space, interesting	steve jobs, imovie,
	iMovie for iPhone 4 was demonstrated.	iphone, #wwdc, imovie, 720p, sensor 3gs, apple, 30fps, gonna, illuminated	wwdc, iphone,
	New iPhone 4 was available in Singapore in July.	iphone, singapore, july, launching, coming, 4g, available, release, #wwdc, early	wifi,

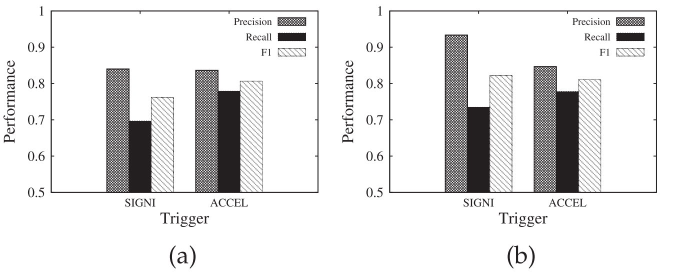


Fig. 14. For different triggers, performances of TD-CM* on (a) San Francisco data and (b) Singapore data.

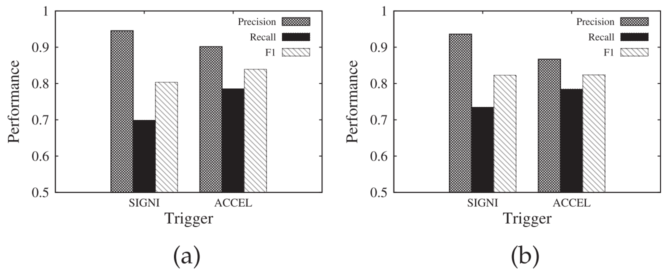


Fig. 15. For different triggers, performances of R-TD-CM* on (a) San Francisco data and (b) Singapore data.

"accelerations of topics", our solution can detect bursty topics in real-time, and present them in finer-granularity.

Appendix A
Infer Topics from Sktech

The tool we use to infer topics from sketch is tensor decomposition [4]. Consider we have the following sketch M_2 and M_3.

$$M_2 = \sum_{k=1}^{K} a_k \cdot \phi_k \otimes \phi_k = \sum_{k=1}^{K} a_k \cdot \phi_k \phi_k^\top$$

$$M_3 = \sum_{k=1}^{K} a_k \cdot \phi_k \otimes \phi_k \otimes \phi_k$$

where $a_k \in \mathbb{R}$, $\phi_k \in \mathbb{R}^N$, $\sum_{w=1}^{N} \phi_{k,w} = 1$. And assume that $a_k \neq 0$ and $\{\phi_k\}_{k=1}^K$ are linear independent. Note that here a_k can be negative, which is weaker than the non-degeneracy assumption in [4]. Project M_3 to a matrix as follows:

$$M_3(\eta) = \sum_k a_k \cdot \phi_k \phi_k^\top \langle \eta, \phi_k \rangle,$$

where $\eta \in \mathbb{R}^N$, is a random vector.

First $rank(M_2) = K$. Short proof here. For any $x \in \mathbb{R}^N$, if $M_2 x = \sum_{k=1}^K a_k \cdot \langle x, \phi_k \rangle \phi_k = 0$, as $\{\phi_k\}_{k=1}^K$ are linear independent, $a_k \cdot \langle x, \phi_k \rangle = 0$, $a_k \neq 0$, so for all the k, $\langle x, \phi_k \rangle = 0$. It means the null space of M_2, $Null(M_2) = span(\{\phi_k\}_{k=1}^K)^\perp$, where $span(\{\phi_k\}_{k=1}^K)^\perp$ is the orthogonal complement of the $span(\{\phi_k\}_{k=1}^K)$. So the dimension of $Null(M_2)$, $nullity(M_2) = N - K$. So $rank(M_2) = N - nullity(M_2) = K$.

As M_2 is a real symmetric matrix, and $rank(M_2) = K$, so that M_2 has k nonzero real eigenvalues $\{\lambda_k\}_{k=1}^K$, and $M_2 = \sum_{k=1}^K \lambda_k \cdot u_k u_k^\top$, where $\{u_k\}_{k=1}^K$ are corresponding eigenvectors, and they are orthonormal. It is easy to show that $Null(M_2) = span(\{u_k\}_{k=1}^K)^\perp$, which means $span(\{\phi_k\}_{k=1}^K) = span(\{u_k\}_{k=1}^K)$.

Whitening. Let $W = \left(u_1/\sqrt{\lambda_1}, \ldots, u_K/\sqrt{\lambda_K}\right)$, we have

$$W^\top M_2 W = I_{K \times K}.$$

Note that W is a complex matrix, because λ_k may be a negative value. So

$$W^\top M_2 W = \sum_{k=1}^{K} a_k \cdot (W^\top \phi_k)(W^\top \phi_k)^\top = I_{K \times K}.$$

It implies for any $k_1 \neq k_2$, $(W^\top \phi_{k_1})^\top (W^\top \phi_{k_2}) = 0$, and $(W^\top \phi_k)^\top (W^\top \phi_k) = a_k^{-1}$.

Whiten $M_3(\eta)$ as follows:

$$T_3 = W^\top M_3(\eta) W = \sum_k a_k \cdot \langle \eta, \phi_k \rangle (W^\top \phi_k)(W^\top \phi_k)^\top.$$

Tensor Power Method

$$T_3(W^\top \phi_k) = \sum_k a_k \cdot \langle \eta, \phi_k \rangle (W^\top \phi_k)(W^\top \phi_k)^\top (W^\top \phi_k),$$
$$= \langle \eta, \phi_k \rangle (W^\top \phi_k).$$

Under the strict condition of distinction, i.e., for any $k_1 \neq k_2$, $\langle \eta, \phi_{k_1} \rangle \neq \langle \eta, \phi_{k_2} \rangle$, T_3 has different eigenvalues $\{\langle \eta, \phi_k \rangle\}_{k=1}^K$ and corresponding eigenvectors $\{W^\top \phi_k\}_{k=1}^K$. Denote the generalized eigenvectors of T_3 as $\{v_k\}_{k=1}^K$, so $v_k = c_k W^\top \phi_k$, where c_k is some complex number.

Reconstruction. As $span(\{\phi_k\}_{k=1}^K) = span(\{u_k\}_{k=1}^K)$, $\phi_k \in span(\{u_k\}_{k=1}^K)$, it means exists some $b_k \in \mathbb{R}^K$ such that $\phi_k = W b_k$. So $v_k = c_k W^\top \phi_k = c_k W^\top W b_k$, $c_k b_k = (W^\top W)^{-1} v_k$, $c_k \phi_k = c_k W b_k = W (W^\top W)^{-1} v_k$. Denote $\tilde{v_k} = W(W^\top W)^{-1} v_k$. As $\sum_{w=1}^N \phi_{k,w} = 1$, $c_k = \sum_{w=1}^N \tilde{v_k}$, i.e., $c_k = 1_N^\top W (W^\top W)^{-1} v_k$.

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

At last, we have,

$$\phi_k = \frac{W(W^\top W)^{-1}v_k}{1_N^\top W(W^\top W)^{-1}v_k},$$

$$a_k = \frac{1}{(W^\top \phi_k)^\top (W^\top \phi_k)}.$$

Acknowledgments

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore, under its International Research Centres in Singapore Funding Initiative. The author would like to thank Professor Alexander J. Smola for suggestions that greatly improved the manuscript.

References

[1] A. Ahmed, Q. Ho, C. H. Teo, J. Eisenstein, A. J. Smola, and E. P. Xing, "Online inference for the infinite topic-cluster model: Storylines from streaming text," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 101–109.

[2] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 1998, pp. 37–45.

[3] F. Alvanaki, S. Michel, K. Ramamritham, and G. Weikum, "See what's enblogue: Real-time emergent topic identification in social media," in *Proc. 15th Int. Conf. Extending Database Technol.*, 2012, pp. 336–347.

[4] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, 2014.

[5] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[6] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 113–120.

[7] T. Brants and F. Chen, "A system for new event detection," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2003, pp. 330–337.

[8] M. Cataldi, L. D. Caro, and C. Schifanella, "Personalized emerging topic detection based on a term aging model," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no 1, 2013, Art. no. 7.

[9] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Proc. Adv. Neural Inform. Process. Syst. 23rd Annu. Conf. Neural Inform. Process. Syst. Meeting*, 2009, pp. 288–296.

[10] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[11] Q. Diao, J. Jiang, F. Zhu, and E. Lim, "Finding bursty topics from microblogs," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, 2012, pp. 536–544.

[12] N. Du, M. Farajtabar, A. Ahmed, A. J. Smola, and L. Song, "Dirichlet-hawkes processes with applications to clustering continuous-time document streams," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 219–228.

[13] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang, "STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream," in *Proc. 31st IEEE Int. Conf. Data Eng.*, 2015, pp. 1561–1572.

[14] T. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Academy Sci. United States Am.*, vol. 101, pp. 5228–5235, 2004.

[15] P. Guttorp, "An introduction to the theory of point processes (D. j. daley and d. vere-jones)," *SIAM Rev.*, vol. 32, no. 1, pp. 175–176, 1990.

[16] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[17] D. He and D. S. P. Jr., "Topic dynamics: An alternative model of bursts in streams of topics," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 443–452.

[18] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 1999, pp. 50–57.

[19] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, "Discovering geographical topics in the twitter stream," in *Proc. 21st World Wide Web Conf.*, 2012, pp. 769–778.

[20] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.

[21] C. Jin, W. Qian, C. Sha, J. Yu, and A. Zhou, "Dynamically maintaining frequent items over a data stream," in *Proc. 12th Int. Conf. Inform. Knowl. Manag.*, 2003, pp. 287–294.

[22] J. Kleinberg, "Bursty and hierarchical structure in streams," *Data Mining Knowl. Discovery*, vol. 7, no. 4, pp. 373–397, 2003.

[23] J. Leskovec, L. Backstrom, and J. M. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 497–506.

[24] C. Li, A. Sun, and A. Datta, "Twevent: Segment-based event detection from tweets," in *Proc. 21st ACM Int. Conf. Inform. Knowl. Manag.*, 2012, pp. 155–164.

[25] M. Mathioudakis and N. Koudas, "Twittermonitor: Trend detection over the twitter stream," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2010, pp. 1155–1158.

[26] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Proc. Human Language Technol.: Conf. North Am. Chapter Assoc. Comput. Linguistics*, 2010, pp. 181–189.

[27] M. Platakis, D. Kotsakos, and D. Gunopulos, "Searching for events in the blogosphere," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 1225–1226.

[28] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: Real-time event detection by social sensors," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 851–860.

[29] E. Schubert, M. Weiler, and H. Kriegel, "Signitrend: Scalable detection of emerging topics in textual streams by hashed significance thresholds," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 871–880.

[30] Y. Takahashi, T. Utsuro, M. Yoshioka, N. Kando, T. Fukuhara, H. Nakagawa, and Y. Kiyota, "Applying a burst model to detect bursty topics in a topic model," in *Proc. Adv. Natural Language Process. 8th Int. Conf.*, 2012, pp. 239–249.

[31] X. Wang, C. Zhai, X. Hu, and R. Sproat, "Mining correlated bursty topic patterns from coordinated text streams," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 784–793.

[32] J. Weng and B. Lee, "Event detection in twitter," in *Proc. 5th Int. Conf. Weblogs Soc. Media*, 2011, pp. 401–408.

[33] W. Xie, F. Zhu, J. Jiang, E. Lim, and K. Wang, "Topicsketch: Real-time bursty topic detection from twitter," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 837–846.

[34] Y. Yang, T. Pierce, and J. G. Carbonell, "A study of retrospective and on-line event detection," in *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 1998, pp. 28–36.

[35] H. Yin, B. Cui, H. Lu, Y. Huang, and J. Yao, "A unified model for stable and temporal topic detection from social media data," in *Proc. 29th IEEE Int. Conf. Data Eng.*, 2013, pp. 661–672.

[36] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Who, where, when and what: Discover spatio-temporal topics for twitter users," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 605–613.

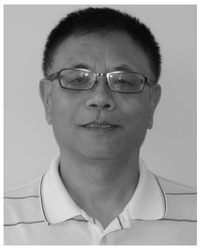
Wei Xie received the bachelor's of mathematics degree from Wuhan University, and the master's of computer software and theory degree from East China Normal University. He is currently working toward the PhD degree in the School of Information Systems, Singapore Management University. His primary research interests include social network mining, as well as text mining.

Feida Zhu received the BS degree in computer science from Fudan University, and the PhD degree in computer science from the University of Illinois, Urbana-Champaign. He is an assistant professor in the School of Information Systems, Singapore Management University. His current research interests include large-scale graph pattern mining and social network analysis, with applications on web, management information systems, business intelligence, and bioinformatics.

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.

Jing Jiang received the BS and MS degrees in computer science from Stanford University, and the PhD degree in computer science from the University of Illinois, Urbana-Champaign. She is an assistant professor in the School of Information Systems, Singapore Management University. Her research centers on text mining, which uses techniques from natural language processing, machine learning and data mining to uncover useful information, and knowledge from textual data.

Ee-Peng Lim received the BSc degree in computer science from the National University of Singapore, and the PhD degree from the University of Minnesota, Minneapolis, in 1994. He is a professor in the School of Information Systems, Singapore Management University. He is the co-director of the Living Analytics Research Center (LARC) jointly established by SMU and Carnegie Mellon University. He is currently an associate editor of the *ACM Transactions on Information Systems* (*TOIS*), *ACM Transactions on the Web* (*TWeb*), *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*), *Information Processing and Management* (*IPM*), *Social Network Analysis and Mining, Journal of Web Engineering* (*JWE*), *IEEE Intelligent Systems, International Journal of Digital Libraries* (*IJDL*), and *International Journal of Data Warehousing and Mining* (*IJDWM*). He was a member of the ACM Publications Board until December 2012. He serves on the Steering Committee of the International Conference on Asian Digital Libraries (ICADL), Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD), and International Conference on Social Informatics (Socinfo). His research interests include social network and web mining, information integration, and digital libraries.



Ke Wang received PhD degree from the Georgia Institute of Technology. He is currently a professor in the School of Computing Science, Simon Fraser University. Before joining Simon Fraser, he was an associate professor at the National University of Singapore. He has taught in the areas of database and data mining. He is particularly interested in combining the strengths of database, statistics, machine learning, and optimization to provide actionable solutions to real life problems. He has published in database, information retrieval, and data mining conferences, including SIGMOD, SIGIR, PODS, VLDB, ICDE, EDBT, SIGKDD, SDM and ICDM. He is currently an associate editor of the *ACM TKDD* journal and he was an associate editor of the *IEEE TKDE* journal and an editorial board member for *Journal of Data Mining and Knowledge Discovery*. He was a general co-chair for the SIAM Conference on Data Mining 2015, and is a general co-chair for the SIAM Conference on Data Mining 2016. He was a PC co-chair for the SIAM Conference on Data Mining 2008 and a PC co-char for the IEEE International Conference on Intelligence and Security Informatics (ISI) 2010. His research interests include database technology, data mining and knowledge discovery, with emphasis on massive datasets, graph and network data, and data privacy.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: Wuhan University. Downloaded on March 20,2020 at 12:01:56 UTC from IEEE Xplore. Restrictions apply.