TopicSketch: Real-time Bursty Topic Detection from Twitter

Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim
Living Analytics Research Centre
Singapore Management University
{*wei.xie.2012, fdzhu, jingjiang, eplim*}*@smu.edu.sg*

Ke Wang
Simon Fraser University
*Singapore Management University**
wangk@cs.sfu.ca

***Abstract*—Twitter has become one of the largest 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 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 modeling and analysis in Twitter, it remains a huge 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 novel 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 can potentially handle hundreds of millions tweets per day which is close to the total number of daily tweets in Twitter and present bursty event in finer-granularity.

***Keywords*-TopicSketch; tweet stream; bursty topic; realtime;**

I. INTRODUCTION

With 200 million active users and over 400 million tweets per day as in a recent report [1], Twitter has become one of the largest information portals which provides an easy, quick and reliable platform for ordinary users to share anything happening around them with friends and other followers. In particular, it has been observed that, in life-critical disasters of societal scale, 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"*.

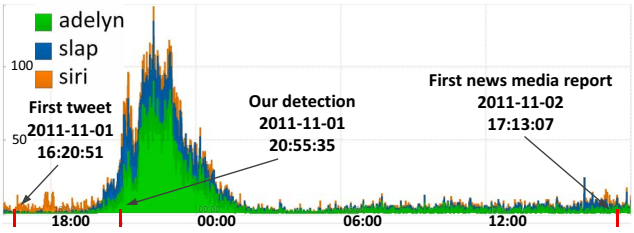

Figure 1. The tweet volume of each of the top three keywords of the topic: "adelyn", "slap" and "siri".

Figure 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 mom's incessant nagging and resorted to physical abuse by slapping her mom 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 to keep track. 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 real-time bursty topic detection on a societal scale?

Unfortunately, this real-time task has not been solved by the existing work on Twitter topic analysis. First of all, 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. Secondly, most prior research works study the topics in Twitter in a retrospective off-line manner, e.g., performing topic modeling, analysis and tracking for all tweets generated in a certain time period [18], [16], [10], [27], [9]. While these findings have offered interesting insight into the topics, it is our belief that the greatest values of Twitter bursty topic detection has yet to be brought out, which is to detect the bursty topics

*This work was done when the author was visiting Living Analytics Research Centre in Singapore Management University.

[1]http://www.techvibes.com/blog/twitter-users-tweet-400-million-times-2012-12-17

[2]http://blog.twitter.com/2011/06/global-pulse.html

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.86

just in time as they are taking place. This real-time task is prohibitively 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 [11] or variational inference [3]. The key research challenge that makes this problem difficult 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 [24] indeed detects events in real-time, it requires pre-defined keywords for the topics.

We propose a new detection framework called TopicSketch. To our best knowledge, this is the first work to perform *real-time* bursty topic detection in Twitter without pre-defined topical keywords. It can be observed from Figure 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 novel data sketch which efficiently maintains at a low computational cost the acceleration of three quantities: the total number of all tweets, the occurrence of each word and the occurrence of each word pair. 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.

Secondly, we proposed dimension reduction techniques based on hashing to achieve scalability and, at the same time, maintain topic quality with proved error bounds.

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 is able to potentially handle over 300 million tweets per day which is almost 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.

II. RELATED WORK

While this work is the first to achieve real-time bursty event detection in Twitter without pre-defined keywords, related work can be grouped into three categories.

Offline. In this category, it is assumed that there is a retrospective view of the data in its entirety. There has been a stream of research studies to learn topics offline from a text corpus, from the standard topic models such as PLSA [14] and LDA [3], to a number of temporal topic models such as [26], [4], [25] and [15]. Since all these models learn topics off-line, they are not able to detect at an early stage the new bursty topics that are previously unseen and just started to grow viral. When it comes to finding bursts from data stream in particular, [18] proposed a state machine to model the data stream, in which bursts appear as state transitions. [16] proposed another solution based on a time-varying Poisson process model. Instead of focusing on arrival rates, [12] reconstructed bursts as a dynamic phenomenon using acceleration and force to detect bursts. Other off-line bursty topic modeling works include most noticeably [10], [27], [9]. While MemeTracker [19] is an influential piece of work which gives an interesting characterisation of news cycle, it is not designed to capture bursty topics on the fly in Twitter-like setting as it is hard to decide what the *meme* of tweets are.

Online. In this category, certain data structure is built based on some inherent granularity defined on the data stream. Detection is made by using the data structure of all data arriving before the detection point but none after. Some works make effort on the online learning of topics [2], [6], [13], while others focus on Topic Detection and Tracking (TDT) such as [1] and [5]. Yet these solutions do not scale to the overwhelming data volume like that of Twitter. In particular, [22] makes use of locality-sensitive hashing (LSH) to reduce time cost. However, even with LSH, the computational cost is huge to calculate, for each arriving tweet, the distances between this tweet and all previous tweets colliding with this tweet in LSH. Twevent [20] is the state-of-the-art system detecting events from tweet stream. The design of Twevent takes an inherent time window of fixed size (e.g., one day) to find bursty segments of tweets, falling short of the full dynamicity essential to the real-time detection task.

Real-time. In this category, time is crucial, so much so that no fixed time window for detection should be assumed. While [24] 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.

Besides, there are also works on finding frequent items from large data stream with small memory such as Count-Min Sketch [8] among others including [21], [7], [17]. Our TopicSketch deals with a very different and harder problem of bursty topics.

III. Solution Overview

A. Problem Formulation

We first formulate our real-time Twitter bursty topic detection problem. In defining a bursty topic, we evaluate two criteria: (I) There has to be a sudden surge of 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 the total number of relevant tweets is neglectable.

Denote $D(t)$ as the set of all tweets generated in the tweet stream up to a given timestamp t. Each tweet $d \in D(t)$ is represented as a bag of words denoted as a vector $\{d(i)\}_{1 \leq i \leq N}$ where $d(i)$ is the number of appearance of word i in d and N is the size of the vocabulary. Each tweet d is associated with the timestamp of its generation denoted as t_d. We use $|d|$ to denote the number of words in tweet d.

We model the tweet stream as a mixture of multiple latent topic streams, where each topic stream is an inhomogeneous Poisson process [23] of a topic. A latent topic T_k is characterized by a fixed distribution p_k over words such that each word in a tweet is drawn from a multinomial distribution of p_k, and we use $\{T_k\}$ to represent the set of all topics. The *rate* of a topic T_k, denoted as $\lambda_k(t)$, is defined such that for any small time slice $[t, t + \mathrm{d}t)$, there is a probability $\lambda_k(t) \cdot \mathrm{d}t$ that a tweet about T_k is generated. The popularity of a topic is measured by the total number of its associated tweets.

A *bursty* topic is a topic where (I) there is a sudden surge of its popularity; and (II) it is reasonable popular in the whole tweet stream in the period when it surges. Given tweet stream $D(t)$, our task in this paper is to detect bursty topics from $D(t)$ as early as possible.

B. Solution Overview

The three primary research challenges here are (I) How to identify the bursty topics, i.e., what are the keywords of the topics, (II) How to detect a bursty topic as *early* as possible, and (III) How to perform the task efficiently in large-scale real-time setting as Twitter.

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 to both challenge (I) and (II) above. In the first step, it maintains as a sketch of the data the acceleration of three quantities: (1) the whole tweet stream, (2) every word, and (3) every pairs 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 solving an optimisation problem. To solve challenge (III), 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.

Figure 2 gives the overview of our proposed TopicSketch framework with five parts: (I) the tweet stream, (II) the data sketch, (III) the monitor which tracks changes in the data sketch and triggers the detection when certain criteria are satisfied, (IV) the estimator which infers the topics, and (V) the reporter which evaluates the bursty topics provided by estimator and reports the detection result. The real-time detection flow of TopicSketch is the following steps: (1) Upon the arrival of each tweet, the sketch is updated, which is an efficient step as detailed in Section IV-A and V-B, (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 Section IV-B and V-C, 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 V-A.

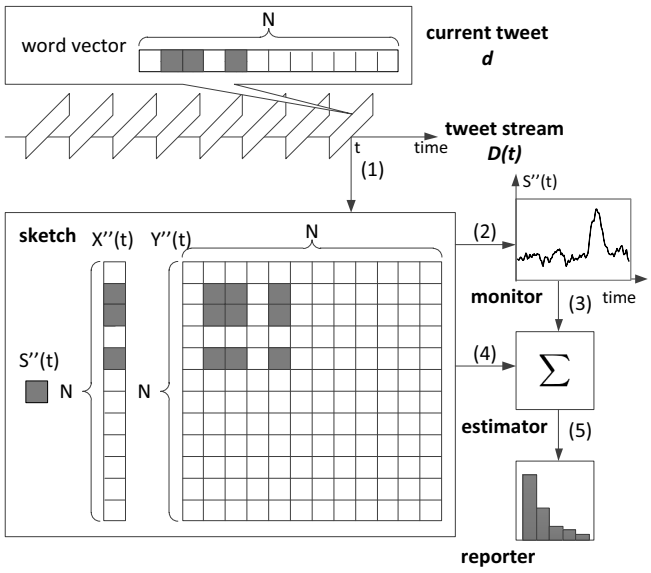

Figure 2. TopicSketch Framework Overview

IV. Sketch-based Topic Model

We first show how TopicSketch is able to detect potential bursty topics early by maintaining a novel data sketch, and

then present how TopicSketch learns the bursty topics based on the data sketch.

A. Sketch

Recall that the popularity of a topic is measured by the total number of relevant tweets. However, it will be too late if we wait till we observe the surge in volume to report the bursty topic. An earlier indicator is the rate of a topic as, mathematically, volume is rate integrated over time. Our idea of early detection is to monitor the acceleration of a topic which, compared against volume and rate, gives an even earlier indicator of the popularity surge. The *acceleration* of a topic T_k, denoted as $a_k(t)$, is defined as the derivative of $\lambda_k(t)$ over time t. However, as $a_k(t)$ is not directly observable for a latent topic, we need to infer $a_k(t)$ from those observable quantities from $D(t)$. In general, the acceleration we monitor for a chosen quantity $\mathbb{Q}(t)$ can be mathematically expressed as

$$\frac{\mathrm{d}^2\mathbb{Q}(t)}{\mathrm{d}t^2} = \mathbb{Q}''(t)$$

In order to achieve both early detection and latent topic identification, we propose to build a data sketch for $D(t)$ and set $\mathbb{Q}(t)$ to be each of the following three quantities to be monitored. Figure 2 gives an illustration. (N is the total number of distinct words.)

(1). $\mathbb{S}''(t)$: The acceleration of the total number of tweets in $D(t)$, i.e.,$\mathbb{Q}(t)$ becomes a scalar denoted as $\mathbb{S}(t)$ such that $\mathbb{S}(t) = |D(t)|$.

(2). $\mathbb{X}''(t)$: The acceleration of each word in the vocabulary, i.e., $\mathbb{Q}(t)$ becomes a N-dimension vector denoted as $\mathbb{X}(t)$ such that $\mathbb{X}_i(t) = \sum_{d \in D(t)} \frac{d(i)}{|d|}$, $(1 \le i \le N)$.

(3). $\mathbb{Y}''(t)$: The acceleration of each pair of words, i.e., $\mathbb{Q}(t)$ becomes a $N \times N$ matrix denoted as $\mathbb{Y}(t)$ such that

$$\mathbb{Y}_{i,j}(t) = \begin{cases} \sum_{d \in D(t)} \frac{d(i)^2 - d(i)}{|d|(|d|-1)} & , \quad i = j \\ \sum_{d \in D(t)} \frac{d(i)d(j)}{|d|(|d|-1)} & , \quad i \neq j \end{cases}$$

$(1 \le i \le N, 1 \le j \le N)$.

These three quantities are chosen because (1) $\mathbb{S}''(t)$ and $\mathbb{X}''(t)$ provides the earliest indicator of popularity surge, (2) $\mathbb{Y}''(t)$ maintains keyword correlation information for bursty topic identification later, and (3) Combined, they can help infer the latent bursty topics (i.e., p_k) as well as the acceleration of each (i.e., $a_k(t)$), as we show next. Notice that all these accelerations are easy to compute and update upon the arrival of every tweet (as shown in Section V), which is critical for scalability in real-time setting.

B. Topic from Sketch

Besides early detection, the sketch also solves the challenge of identifying the bursty topics. We envision a space of latent topics, and we model the tweet stream as a mixture of multiple latent topic streams. While we set no limit on the total number of latent topics, we assume that at any time stamp t, there is an upper bound K on the total number of *active* topics T_k whose rate $\lambda_k(t)$ is greater than zero. Therefore, at any time stamp t, we are only interested in discovering the K active latent topics that are bursty. To identify bursty topics from the data sketch, we first show some useful properties of the three accelerations.

By the superposition property of inhomogeneous Poisson process [23], the whole tweet stream, which is a mixture of multiple inhomogeneous processes of topics, is also an inhomogeneous Poisson process itself. Its rate function is $\sum_{k=1}^{K} \lambda_k(t)$. We can simply use $\mathbb{S}'(t) = \frac{\mathrm{d}\mathbb{S}(t)}{\mathrm{d}t}$ to estimate it. Further we have the following equation,

$$\mathbb{S}''(t) = \sum_{k=1}^{K} a_k(t) \tag{1}$$

Then by the property of linear combinations of expectation, it is easy to derive

$$E[\mathbb{X}''(t)] = \sum_{k=1}^{K} a_k(t) \cdot p_k \tag{2}$$

$$E[\mathbb{Y}''(t)] = \sum_{k=1}^{K} a_k(t) \cdot p_k \cdot p_k^T \tag{3}$$

where p_k is the vector representing the distribution over words of topic T_k.

The equations 1, 2 and 3 imply that we can infer the topics $\{T_k\}$ and their acceleration from the sketch. At time t, we can estimate the parameters $\{p_k\}$ and $\{a_k(t)\}$ from the data sketch as follows. We seek such $\{p_k\}$ and $\{a_k(t)\}$ that satisfy Equation 1 and minimize the differences between the observed values and the expectations as in Equation 2 and Equation 3. Denote the weights for Equation 2 and 3 as $w_X > 0$ and $w_Y > 0$ respectively. To estimate $\{p_k\}$ and $\{a_k(t)\}$, we only need to solve the following optimization problem.

minimize

$$f = w_X \cdot e_X + w_Y \cdot e_Y \tag{4}$$

s.t.

$$\sum_{k=1}^{K} a_k(t) = \mathbb{S}''(t) \tag{5}$$

$$\sum_{i=1}^{N} p_{k,i} = 1, 1 \le k \le K \tag{6}$$

$$p_{k,i} \ge 0, 1 \le k \le K, 1 \le i \le N \tag{7}$$

where

$$e_X = \sum_{i=1}^{N} (\sum_{k=1}^{K} a_k(t) \cdot p_{k,i} - \mathbb{X}''_i(t))^2 \quad (8)$$

$$e_Y = \sum_{i=1}^{N} \sum_{j=1}^{N} (\sum_{k=1}^{K} a_k(t) \cdot p_{k,i} \cdot p_{k,j} - \mathbb{Y}''_{i,j}(t))^2 \quad (9)$$

e_X and e_Y in objective function 4 are the summed square error in Equation 2 and Equation 3 respectively. Condition 5 is indeed Equation 1. Weights w_X and w_Y are to be tuned empirically as the variance of $\mathbb{X}''(t)$ and $\mathbb{Y}''(t)$ are different. More details on setting the weights are discussed in the experiment part of Section VI.

V. 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.

A. 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 VI-A). This results not only in an enormous data sketch (recall $\mathbb{Y}''(t)$ in the sketch is an $N \times N$ matrix) but also an optimization problem of very high dimensions, i.e. $O(N \cdot K)$.

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 VI-A) to solve the optimization problem efficiently.

Instead of keeping a set of active words, we propose a novel solution by hashing 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 $O(B^2)$ and the number of dimensions for the optimization problem is reduced to $O(B \cdot K)$, which are significantly smaller than $O(N^2)$ and $O(N \cdot K)$ as in the original problem. After hashing, what we obtain is the distribution over buckets, rather than the distribution over words, which means we would need to recover the probabilities of words from the probabilities of buckets. To solve this problem, the observation is that, for the word distribution of bursty topics, we care only the top words with highest probabilities which represent the bursty topic. Therefore we adapt the count-min algorithm in [8], [17] to our setting, which can dynamically maintain frequent items over data streams. The difference is that in our setting we want to maintain words with high probability in a distribution rather than items with high frequency in a data stream. However, the underling logic works the same for both settings, which is the following: If we use H hash functions, instead of just one, to map each word, the probability that two top words of a topic fall into the same bucket for all these H hash functions is extremely small. More importantly, if there is only one word with dominantly high probability in a bucket, we can then use the probability of the bucket to approximate the probability of the high-probability word.

In particular, it works as follows. Assume we have H hash functions $(\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_H)$ which map words to buckets $[1...B]$ uniformly and independently. For a topic T_k with word distribution p_k, and each hash function $\mathcal{H}_h, 1 \leq h \leq H$, we can estimate the distribution over buckets $\{p_{k,j}^{(h)} = \sum_{i|\mathcal{H}_h(i)=j} p_{k,i}\}_{j=1}^{B}$ for all the hash functions. Then we use count-min algorithm to estimate the probability of each word i as $min_{1\leq h\leq H}\{p_{k,\mathcal{H}(i)}^{(h)}\}$, and return the words of high probability $\{i|min_{1\leq h\leq H}\{p_{k,\mathcal{H}(i)}^{(h)}\} \geq s\}$, where s is a probability threshold, e.g., 0.02. We also maintain a set of active words, so that we estimate the probability of words only in this set rather than all the words in the vocabulary. 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 [17].

The sketch after dimension reduction is illustrated in Figure 3. There is no change for $\mathbb{S}''(t)$. When a word falls into different buckets for different hash functions, we maintain H vectors $\{\mathbb{X}''(t)^{(h)}\}_{h=1}^{H}$ for $\mathbb{X}''(t)$ and H matrices $\{\mathbb{Y}''(t)^{(h)}\}_{h=1}^{H}$ for $\mathbb{Y}''(t)$.

To estimate the distribution over buckets $\{p^{(h)}\}_{j=1}^{B}$, it is sufficient to change Equation 8 and Equation 9 as follows.

$$e_X = \sum_{h=1}^{H} \sum_{i=1}^{B} (\sum_{k=1}^{K} a_k \cdot p_{k,i}^{(h)} - \mathbb{X}_i''^{(h)})^2 \quad (10)$$

$$e_Y = \sum_{h=1}^{H} \sum_{i=1}^{B} \sum_{j=1}^{B} (\sum_{k=1}^{K} a_k \cdot p_{k,i}^{(h)} \cdot p_{k,j}^{(h)} - \mathbb{Y}_{i,j}''^{(h)})^2 \quad (11)$$

Accordingly Condition 6 and 7 should be adjusted as follows.

$$\sum_{i=1}^{B} p_{k,i}^{(h)} = 1, 1 \leq k \leq K, 1 \leq h \leq H \quad (12)$$

$$p_{k,i}^{(h)} \geq 0, 1 \leq k \leq K, 1 \leq i \leq B, 1 \leq h \leq H \quad (13)$$

After the dimension reduction, the memory cost for the sketch is $O(H \cdot B^2)$, and the number of dimension for the optimization problem is $O(H \cdot B \cdot K)$, which are small enough to be practically feasible.

B. Efficient Sketch Maintenance

We adopt an idea similar as EMA (Exponential Moving Average) in [12] to estimate the rate. The difference is that [12] processes discrete time series data, while we process a continuous tweet stream, i.e. tweets can arrive at any time point. Denote $D(t)$ as $\{d_1, d_2, ..., d_{|D(t)|}\}$, such that $t_{d_1} \leq$

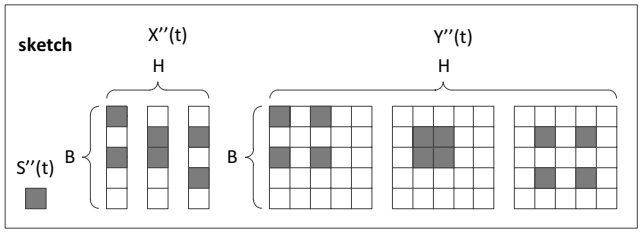

Figure 3. New Sketch after Dimension Reduction

$t_{d_2} \leq ...t_{d_{|D(t)|}} \leq t$. And set $t_{d_0} = 0$. Equation 14 below estimates the rate. ΔT here is the smooth factor. The larger the ΔT, the more smooth the rate, but less reflective of the recent information though.

$$\mathbb{S}'_{\Delta T}(t) = \sum_{i=1}^{|D(t)|} \frac{e^{\frac{(t_{d_i}-t)}{\Delta T}}}{\Delta T} \tag{14}$$

For any time point t between $(t_{d_{i-1}}, t_{d_i}]$, we can update the current rate incrementally by Equation 15.

$$\mathbb{S}'_{\Delta T}(t) = \begin{cases} \mathbb{S}'_{\Delta T}(t_{d_{i-1}}) \cdot e^{\frac{(t_{d_{i-1}}-t)}{\Delta T}}, & t \in (t_{d_{i-1}}, t_{d_i}) \\ \mathbb{S}'_{\Delta T}(t_{d_{i-1}}) \cdot e^{\frac{(t_{d_{i-1}}-t)}{\Delta T}} + \frac{1}{\Delta T}, & t = t_{d_i} \end{cases} \tag{15}$$

Similar to MACD (Moving Average Convergence / Divergence) in [12], we use Equation 16 to estimate the acceleration.

$$\mathbb{S}''_{\Delta T_1, \Delta T_2}(t) = \frac{\mathbb{S}'_{\Delta T_1}(t) - \mathbb{S}'_{\Delta T_2}(t)}{\Delta T_2 - \Delta T_1} \tag{16}$$

Same as ΔT, ΔT_1 and ΔT_2 are the smooth factors. The computational cost for maintaining any acceleration is therefore $O(1)$.

Besides, we developed a lazy maintenance technique to efficiently maintain each acceleration in $\mathbb{X}''(t)$ and $\mathbb{Y}''(t)$, reducing the number of accelerations to update from $O(N^2)$ to $O(H \cdot |d|^2)$. The details are omitted here due to space limit.

C. Topic Inference

To solve the optimization problem in Section IV-B, we use a gradient-based method to optimize the objective function f over the parameters $\{p_{k,i}^{(h)}\}$ and $\{a_k\}$. Although the number of dimensions is reduced, it is still significant for optimization. To cope with that, we update $p_{k,i}^{(h)}$, a_k separately, rather than update them together, so that some updates can be performed in parallel. We present here a coordinate ascent method for the optimization. Denote $\{a_k\}_{k=1}^K$ as vector $\boldsymbol{a}$, $\{p_{k,i}^{(h)}\}_{i=1}^B$ as vector $\boldsymbol{p}_k^{(h)}$. We can derive their coordinate-wise gradients : $\frac{\partial f}{\partial \boldsymbol{a}}$, $\frac{\partial f}{\partial \boldsymbol{p}_k^{(h)}}$, and their second derivatives : $\frac{\partial^2 f}{\partial \boldsymbol{a} \partial \boldsymbol{a}^T}$, $\frac{\partial^2 f}{\partial \boldsymbol{p}_k^{(h)} \partial \boldsymbol{p}_k^{(h)T}}$. After initializing $\boldsymbol{a}$ and $\boldsymbol{p}_k^{(h)}$, we can iteratively update them by using Newton-Raphson approach. As when $\boldsymbol{a}$ is fixed, $\boldsymbol{p}_k^{(h)}$ are independent for different h, we can update them in parallel. Since there are linear constraints for both $\boldsymbol{a}$ and $\boldsymbol{p}_k^{(h)}$, the Newton-Raphson approach is further adapted so that $\boldsymbol{a}$ and $\boldsymbol{p}_k^{(h)}$ would be correctly updated. We check whether the maximum number of iterations is reached or parameters converge to decide whether the stop criterion is satisfied. An overview of this optimization procedure is given in Table I.

while stop criterion is not satisfied:
  for $h = 1...H$ (in parallel)
    for $k = 1...K$
      fixing $\boldsymbol{a}$ and $\{\boldsymbol{p}_{k'}^{(h)}\}_{k' \neq k}$, use Newton-Raphson approach to find best $\boldsymbol{p}_k^{(h)}$ based on $\frac{\partial f}{\partial \boldsymbol{p}_k^{(h)}}$ and $\frac{\partial^2 f}{\partial \boldsymbol{p}_k^{(h)} \partial \boldsymbol{p}_k^{(h)T}}$
    endfor
  endfor
  fixing $\{\boldsymbol{p}_k^{(h)}\}_{k=1}^K$, use the Newton-Raphson approach to find the best $\boldsymbol{a}$ based on $\frac{\partial f}{\partial \boldsymbol{a}}$ and $\frac{\partial^2 f}{\partial \boldsymbol{a} \partial \boldsymbol{a}^T}$
endwhile

Table I
THE TOPIC INFERENCE ALGORITHM

VI. EVALUATION

In this section, we present the evaluation of our TopicSketch system for both efficiency and effectiveness. We use a Twitter data set which contains 3,165,479 users. These users are obtained by a snowball-style crawling starting from a seed set of Singapore local celebrities and active users and tracing their follower/followee links up to two hops. We crawled all their tweets. In this evaluation we use the subset of tweets between April 1, 2013 and April 30, 2013 to simulate a live tweet stream, which contains 32,479,134 tweets. Some spam accounts are filtered manually. We implemented our solution in Java 1.7 using 64-bit addressing, and executed on multiple cores of an Intel Xeon 3.06 GHz machine. We only evaluated our solution after dimension reduction as the dimension of the original solution presented in Section IV-B is too high to be solved practically. The number of buckets B used in the universal hashing has been empirically set to 300, and the number of hash functions H to 5 for a good balance between efficiency and effectiveness.

A. 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.

1) Sketch Maintenance: In our experiment, we set the smooth factor as : ΔT_1=15 minutes, ΔT_2 = 5 minutes. According to the analysis in Section V-B, the complexity for maintaining the sketch is $O(H \cdot |d|^2)$. It is not hard to maintain the sketch in parallel on multiple cores or machines. We maintain the sketch on multiple cores on a single machine, which can be extended easily to multiple machines for better performance. We partition the job into H pieces, each in charge of one hash function $\mathcal{H}_h$, i.e. maintaining $\mathbb{X}''(t)^{(h)}$ and $\mathbb{Y}''(t)^{(h)}$. We build a thread pool, and submit H jobs to the thread pool for each arriving tweet. To evaluate the throughput of the sketch maintenance and its scalability, we set the number of threads in the thread pool from 1 to 6 respectively. Figure 4(a) shows the throughput for thread pool of different sizes. Notice that when the number of the threads is 1, the throughput is 4,289 tweets per second (over three hundred millions tweets per day), which is roughly the total number of tweets generated daily in the whole Twitter network. In contrast, we find in our experiments that it would take many days to process the data of this scale with LDA-based methods. Also observe that the throughput increases as the number of threads increases. As we can see the increase in our experiment is not linear. In fact, as H = 5, the total number of jobs is 5, which are assigned to the threads in thread pool, the ideal ratio of time per tweet is 5 : 3 : 2 : 2 : 1 : 1. That is why the throughputs for 3 threads and 4 threads, as well as for 5 threads and 6 threads are almost the same. However, we can see a slight drop when the number of threads increases to 6. One possible explanation is because of the additional coordination cost for one additional thread.

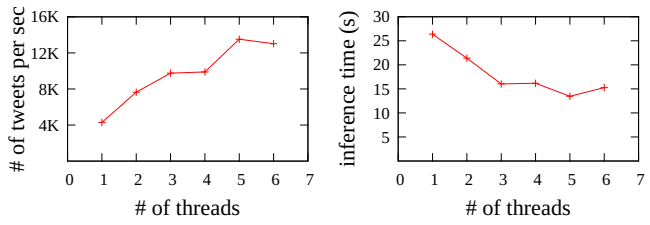

Figure 4. (a) Throughput and (b) Inference time

2) Topic Inference: We empirically set w_X = 0.01 and w_Y = 1 as we have observed that the variance of $\mathbb{X}''$ is larger than that of $\mathbb{Y}''$. We set the number of topics K to a small number of 5. As described in Section V-C, we designed a parallel algorithm to infer the parameters $\{p_{k,i}^{(h)}\}$ and $\{a_k\}$, and implemented it on multiple cores on a single machine. In particular, we built a thread pool and submitted H jobs to the thread pool. Each job is in charge of updating parameters $\{\boldsymbol{p}_k^{(h)}\}_{k=1}^K$ for one hash function $\mathcal{H}_h$ using Newton-Raphson approach. We set the initial values of $p_{k,i}^{(h)}$ to $\frac{1}{B}$ and a_k to $\frac{\mathbb{S}''}{K}$. In addition, we set the maximum number of iterations for the Newton-Raphson approach to 25, the maximum number of outer loops to 50, and the size of thread pool from 1 to 6. The performance of the algorithm is shown in Figure 4(b). The result shows that although the sequential version features an affordable running time (less than half minute), parallel version provides significant improvement (15 seconds with 5 threads). Same as Figure 4(a), the performance drops slightly when the number of threads is increased to 6, which may due to the additional coordination cost.

B. Effectiveness Evaluation

To evaluate the effectiveness of our solution, we compare TopicSketch against both Twitter's official trending topic list and the state-of-the-art Twitter event detection system Twevent [20]. While TopicSketch detects bursty topics real-time, Twevent reports events on a daily basis. To compare the two, in Section VI-B1 we list the topics detected by both and discuss in detail the differences. When comparing against Twitter's own trending topics in Section VI-B2, we present the timestamps of detected bursty topics from TopicSketch.

1) Comparison with Twevent: To compare with Twevent, we used the same dataset as used in the original paper [27], which is a collection of 4,331,937 tweets published by 19,256 unique Singapore based users (according to the user profile information). As there is no ground truth along with this dataset, Twevent was evaluated in [20] by precision and recall based on manual labelling. Instead of simply comparing precision and recall with theirs based on our own manual labelling, which is hardly objective, we present in Table II all the events detected by both algorithms[3] between June 7, 2010 to June 12, 2010, in which period several big events happened, including Apple WWDC 2010, MTV Movie Awards 2010, and FIFA World Cup 2010, and compare the differences between the results. We manually group together sub-events belonging to a single larger event.

We have the following observations from the result comparison. **First, for all topics with significant bursts captured by both algorithms, TopicSketch provides temporally-ordered sub-events that are more descriptive of the corresponding single event detected by Twevent**. Take the event of MTV Movie Awards for example, Twevent provides these segments: *mtv movie awards*, *mtv*, *new moon*, *twlight*, *robe*, which can represent this event and also the sub-event "The Twilight Saga : New Moon" got the Best Movie Award. TopicSketch detected three bursty topics "*stewart, kristen, female, mtv*", "*sandra, bullock, mtv, movie*" and "*movie, moon, mtv, awards*", in which each represents a sub-event listed in Table II. **Second, TopicSketch is able to detect events with bursts over shorter duration**

[3]We referred the results listed in Table 2 from [20]. And the repeated events were filtered out.

Date	Event	Sub-Event	TopicSketch	Twevent
7	MTV Moive Awards 2010	Kristen Stewart won the Best Female Performance	stewart,kristen,female,mtv	mtv movie awards, mtv, new moon, twilight, robe
		Sandra Bullock won the Generation Award	sandra,bullock,mtv,movie	
		Best Movie Award :"The Twilight Saga: New Moon"	movie,moon,mtv,awards	
	Super Junior's Yesung(@shfly3424) created his Twitter account.		yesung, twitter, @shfly3424	**None**
	Fans celebrated 3 year anniversary for boy band "F.T. Island" in Twitter.		f3island, love, <3	**None**
	Steve Jobs released iPhone 4 during WWDC2010	Farmville client for iPhone 4 was demonstrated.	#wwdc, iphone, farmville	steve jobs, imovie, wwdc, iphone, wifi
		Retina display of iPhone 4 was introduced.	iphone,4,#wwdc,display,retina	
		iMovie for iPhone 4 was demonstrated.	iphone, 4, imovie, #wwdc	
		New iPhone 4 was available in Singapore in July.	iphone, 4, singapore, july	
8	Fans asked pop musician Justin Bieber to follow them in Twitter		@justinbieber, follow, please	**None**
	Fans celebrated 5 year anniversary for boy band "SS501" in Twitter		#5yrss501, ss501, #5yearss501	**None**
	The music video "Alejandro" by Lady GaGa was premiered		alejandro, video, gaga	lady gaga, alejandro music video, gaga, mv
9	Korean pop singer Eunhyuk(@allrisesilver) posted his new photo and fans said good afternoon to him.		afternoon, @allrisesilver, http://twitpic.com/1v6sc2	**None**
	Fans tried to trend hashtag #loveisalexander(Alexander is a member of Korean pop boy band U-KISS).		#loveisalexander, <3, @alexander_0729	**None**
	A number of users complained they could not use twitter due to over-capacity. A logo with whale is usually used to denote over-capacity.		**None**	twitter, whale, stupid, capacity, over again
	The season finale of American TV series Glee was broadcasted on June 8, 2010.		glee, yeah, watching	watching glee,glee,season season finale,channel
10	Korean pop boy group Infinite (consists of 7 members) had a performance.		#infinite7, huh	**None**
	Korean actor Ok Taec-Yeon(@taeccool) opened twitter account.		@taeccool,twitter,taecyeon,taec	**None**
	The movie The Karate Kid was released on June 10, 2010 in Singapore.		**None**	karate kid, movie watch movie
	Super Junior's Yesung posted a photo about his pet turtles.		**None**	yesung, tweeted
11	Super Junior was performing "Bonamana" on Music Bank.		super, junior, bonamana	**None**
	SS501 won the K-Chart on Music Bank.		ss501, won, congrats	**None**
	South Africa vs Mexico in World Cup 2010.	Match began.	south,world,cup,africa,mexico	south africa, vs mexico, mexico, goal, first goal
		South Africa first goaled. (1-0)	africa,south,goal,mexico,1-0	
		Mexico goaled. (1-1)	mexico, goal, 1-1	
		At last draw.	mexico, africa, draw, south	
	Uruguay vs France in World Cup 2010.		**None**	uruguay va france, uruguay, france, vs
12	South Korea vs Greece in World Cup 2010.	Match began.	korea, south, greece, #kor	south korea, greece, korea vs greece, korea won, korea
		South Korea first goaled.	korea, goal, scored, 1	
		Park Ji-Sung from Korea goaled.	korea, goal, ji, 2-0	
		South Korea won the match.	korea, won, #kor, win	
	Argentina vs Nigeria in World Cup 2010.	Argentina first goaled.	argentina, goal, 1-0	arg, argentina, nigeria, argentina vs nigeria,messi
		Argentina won the match.	argentina, 1-0, nigeria, won	
	England vs USA in World Cup 2010.	Match began.	england, vs, usa, match	usa, england, eng, vs
		Steven Gerrard from England goaled.	gerrard, england, steven, goal	
		GoalKeeper Robert Green didn't catch a ball.	green, robert, wtf	
		USA goaled.	usa, england, goal, 1	

Table II
LIST OF EVENTS DETECTED BY TOPICSKETCH AND TWEVENT

which are missed by Twevent. For example, the event of fans celebrating 5-year anniversary for boy band "SS501" in Twitter. Twevent missed this event because this event, which appeared as a burst in about 2 hours and then disappeared, does not constitute a burst significant enough when considered in a whole-day period. **Third, TopicSketch would miss events with no significant bursts**. For instance, the match between Uruguay and France in World Cup 2010 was detected by Twevent, but missed by TopicSketch. By checking the tweets of that day, we found that this match was hold at about 2am in Singapore time zone and caused no significant burst. In case study part of Section VI-C, we give more detailed examples to explain the reasons behind these observations.

2) Comparison with Twitter trending topic: The trending topics provided by Twitter API are usually represented by single words, hashtags and phrases. Our list of trending topics is obtained at a frequency of every 5 minutes from the beginning of 2013. To demonstrate the difference between the trending topics as presented by Twitter and the bursty topics we are interested in, which are two related yet different concepts, we focus on the day of April 8, 2013, when the former British Prime Minister Margaret Thatcher died. We enumerated all the trending topics of Twitter for this day. By verifying bursty events and filtering out those that are not (including those all-time popular topics), we found

Event	First detected by TopicSketch	First appeared in Twitter
The popular program WrestleMania was discussed.	None	#wrestlemania **(14:34:12)**
Fans asked Luke Brooks (@luke_brooks) to follow them.	@luke_brooks, #followmeluke, follow (19:06:51)	#followmeluke **(18:54:12)**
The sudden demise of Margaret Thatcher.	thatcher, margaret **(20:05:51)**	margaret thatcher (20:49:12)

Table III
LIST OF EVENTS DETECTED BY TOPICSKETCH AND TWITTER ON APRIL 8, 2013

three events as shown in Table III. In Figure 5, we show the number of tweets per minute which contain the keyword of each event and the time stamp when TopicSketch and Twitter each detected them.

Our observation is that: **the more bursty the event is, the better TopicSketch performs.** As there was no obvious burst, TopicSketch missed the event of *wrestlemania*. For the event of *followmeluke*, Twitter is about 10 minutes faster than TopicSketch. However, for the big breaking event of *thatcher*'s demise, TopicSketch is about 40 minutes faster than Twitter, and only about 15 minutes later than the first relevant tweet in our dataset.

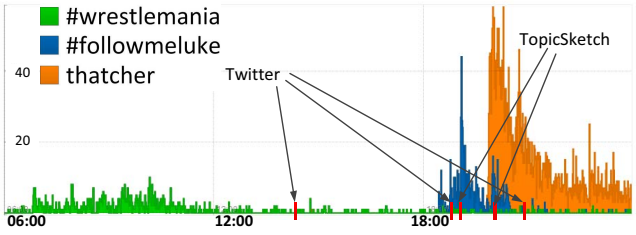

Figure 5. Topics detection from TopicSketch and Twitter.

C. Case Studies

In this section, we discuss the advantages and limitations of TopicSketch through some illustrative examples.

Finer-granularity Event Description. We use the example of Apple WWDC 2010 to demonstrate 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 II, TopicSketch not only detected the big event of Apple WWDC 2010 as a whole, it also detected a sequence of sub-events. Along timeline, we show in Figure 6(a) the number of tweets per minute which contain a keyword of each sub-event, in Figure 6(b) the number of tweets per minute which contain the keywords of this event – wwdc and #wwdc. From this figure we can see the duration of the burst of "wwdc" is quite long for one day period, which is a big trend. So it is significant enough to be detected by both TopicSketch and Twevent. Compared with "wwdc", the duration of the burst of each sub-event is relatively short, which is about 15 minutes. The keywords such as "display" appeared and disappeared in a short time period and it is not significant enough considered against a longer time window to be detected by Twevent. As the sketch captures the acceleration of the tweet stream which reflects real-time data dynamics, TopicSketch successfully detected them. In particular, each peak in the figure which triggers our system indeed corresponds to a highlight of the WWDC event, which are some of the features as they were being announced, and from which we can tell those new features of iPhone that users find more interesting than others.

Bursty Topic vs Continual Topic. As shown in Table II, on June 10, TopicSketch detected the event of the performance of Infinite while Twevent missed it. In the meantime, Twevent detected the event of the release of "The Karate Kid", while TopicSketch missed it. In Figure 6(c) we show the number of tweets per minute which contain the keywords of these two event – *#infinite7* and *karate*. Form this figure, we can see that for *karate*, there was no obvious burst on the timeline as it was being continually discussed. In contrast, for *#infinite7*, we can see a major burst over a few hours along the timeline. It is clear that, due to its daily-base detection, Twevent is good at detecting events that are continually discussed over a long period of time, but may miss events with shorter bursts. On the other hand, one would arrive almost the opposite conclusion for TopicSketch.

Spam Detection It is interesting to note that TopicSketch could help detect spam by surfacing bursty topics with regular appearing patterns. Figure 6(d) illustrates the volume of tweets which contain a spam url in one day. A burst roughly every 4 hours can be easily identified. Although TopicSketch is not designed for spam detection, spam suspects, once detected, can be easily verified by further tracking.

VII. 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 novel concept of "Sketch", 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. Compared

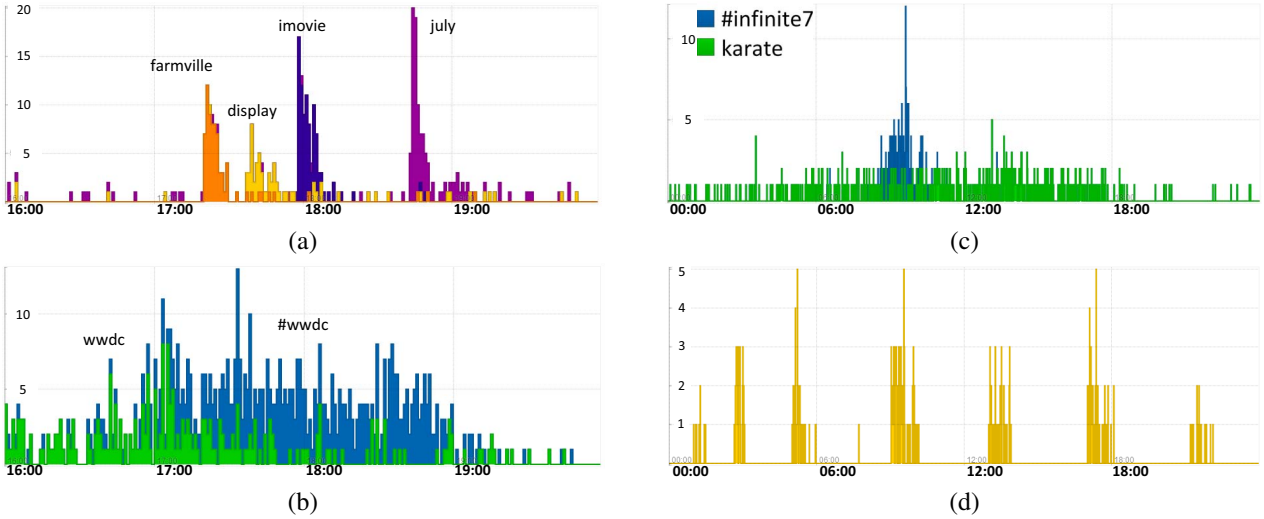

Figure 6. Case studies. (a)-(b) Apple WWDC 2010; (c) events *infinite7* and *karate*; (d) detected bursty topic created by spam.

with existing event detection system, our experiments have demonstrated the superiority of TopicSketch in detecting bursty topics in real-time.

ACKNOWLEDGMENT

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

REFERENCES

[1] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*, pages 37–45, 1998.

[2] L. AlSumait, D. Barbará, and C. Domeniconi. On-line lda: adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, 2008.

[3] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[4] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120, 2006.

[5] T. Brants, F. Chen, and A. Farahat. A system for new event detection. In *SIGIR*, pages 330–337, 2003.

[6] K. R. Canini, L. Shi, and T. L. Griffiths. Online inference of topics with latent dirichlet allocation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 65–72, 2009.

[7] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 296–306, 2003.

[8] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[9] Q. Diao, J. Jiang, F. Zhu, and E.-P. Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544, 2012.

[10] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st international conference on Very large data bases*, pages 181–192, 2005.

[11] T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[12] D. He and D. Parker. Topic dynamics: an alternative model of bursts in streams of topics. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 443–452, 2010.

[13] M. D. Hoffman, D. M. Blei, and F. Bach. Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864, 2010.

[14] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[15] L. Hong, B. Dom, S. Gurumurthy, and K. Tsioutsiouliklis. A time-dependent topic model for multiple text streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 832–840, 2011.

[16] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 207–216, 2006.

[17] C. Jin, W. Qian, C. Sha, J. Yu, and A. Zhou. Dynamically maintaining frequent items over a data stream. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 287–294, 2003.

[18] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[19] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, 2009.

[20] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 155–164, 2012.

[21] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357, 2002.

[22] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *HLT-NAACL*, pages 181–189, 2010.

[23] S. Ross. *Stochastic processes*. Wiley, New York, 1996.

[24] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860, 2010.

[25] C. Wang, D. Blei, and D. Heckerman. Continuous time dynamic topic models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 579–586, 2008.

[26] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433, 2006.

[27] J. Weng and B.-S. Lee. Event detection in twitter. *Proc. of ICWSM*, 2011.