Identifying Event-related Bursts via Social Media Activities

Wayne Xin Zhao[†], Baihan Shu[†], Jing Jiang[‡], Yang Song[†], Hongfei Yan[†*] and Xiaoming Li[†]
[†]School of Electronics Engineering and Computer Science, Peking University
[‡]School of Information Systems, Singapore Management University
{batmanfly,baihan.shu,yhf1029}@gmail.com,ysong@pku.edu.cn
jingjiang@smu.edu.sg, lxm@pku.edu.cn

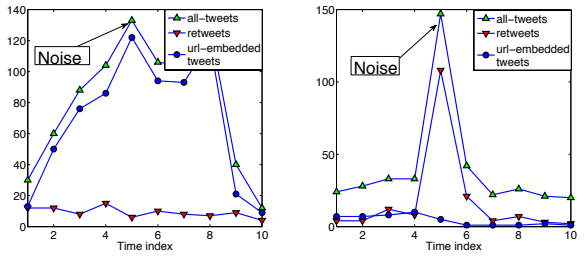
Abstract

Activities on social media increase at a dramatic rate. When an external event happens, there is a surge in the degree of activities related to the event. These activities may be temporally correlated with one another, but they may also capture different aspects of an event and therefore exhibit different bursty patterns. In this paper, we propose to identify event-related bursts via social media activities. We study how to correlate multiple types of activities to derive a global bursty pattern. To model smoothness of one state sequence, we propose a novel function which can capture the state context. The experiments on a large Twitter dataset shows our methods are very effective.

1 Introduction

Online social networks (e.g., Twitter, Facebook, Myspace) significantly influence the way we live. Activities on social media increase at a dramatic rate. Millions of users engage in a diverse range of routine activities on social media such as posting blog messages, images, videos or status messages, as well as interacting with items generated by others such as forwarding messages. When an event interesting to a certain group of individuals takes place, there is usually a surge in the degree of activities related to the event (e.g., a sudden explosion of tweets). Since social media activities may indicate the happenings of external events, can we leverage on the rich social media activities to help identify meaningful external events? This is the research problem we study in this paper. By external events, we refer to real-world events that happen external to the online space.

*Corresponding author.



(a) Query="Amazon." (b) Query="Eclipse".

Figure 1: The amount of activities within a 10-hour window for two queries. Three types of activities are considered: (1) posting a tweet (upward triangle), (2) retweet (downward triangle), (3) posting a URL-embedded tweet (excluding retweet) (filled circle). As explained in Table 1, both bursts above are noisy.

Mining events from text streams is usually achieved by detecting bursty patterns (Swan and Allan, 2000; Kleinberg, 2003; Fung et al., 2005). However, previous work has mostly focused on traditional text streams such as scientific publications and news articles. There is still a lack of systematic investigations into the problem of identifying event-related bursty patterns via social media activities. There are at least two basic characteristics of social media that make the problem more interesting and challenging.

First, social media involve various types of activities taking place in real time. These activities may be temporally correlated with one another, but they may also capture different aspects of an event and therefore exhibit different bursty patterns. Most of previous methods (Swan and Allan, 2000; Kleinberg, 2003; Fung et al., 2005) deal with a single type of textual activities. When applied to social media, they oversimplify the complex nature of online so-

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1466–1477, Jeju Island, Korea, 12–14 July 2012. ©2012 Association for Computational Linguistics

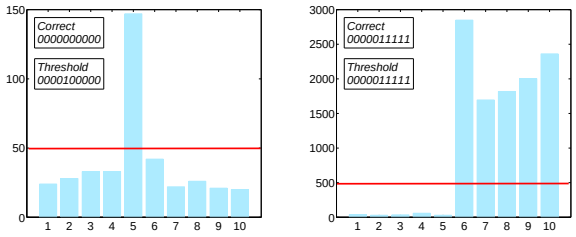
Bursty Activity	Time	# in $\mathcal{S}_r$	# in $\mathcal{S}_u$	# in $\mathcal{S}_t$	Noisy?
$\mathcal{S}_r, \mathcal{S}_t$ See Fig. 1(b)	23:00∼23:59, Nov. 23, 2009	**108**	5	147	**Y**
	[Query=**eclipse**] major bursty reason: The tweet from Robert Pattinson "@twilight: from rob cont . - i hope you are looking forward to **eclipse** as much as i am ." has been retweeted many times.				
$\mathcal{S}_u, \mathcal{S}_t$ See Fig. 1(a)	07:00∼07:59, Jul. 25, 2009	6	**122**	133	**Y**
	[Query=**Amazon**] major bursty reason: Advertisement tweets like "@fitnessjunkies **amazon**.com deals : http://tinyurl.com/lakz3h." have been posted many times.				
$\mathcal{S}_t, \mathcal{S}_u, \mathcal{S}_r$	09:00∼9:59, Oct. 9, 2009	1562	423	2848	**N**
	[Query=**Nobel**] major bursty reason: The news "Obama won **Nobel** Peace Prize" flood Twitter.				

Table 1: Examples of bursts. The first two bursts are judged as noise since they do not correspond to any meaningful external events. In fact, the reasons why a burst appears in social media can be quite diverse. In this paper, we only focus on event-related bursts. $\mathcal{S}_t$ denotes posting a tweet, $\mathcal{S}_u$ denotes posting a url-embedded tweet, and $\mathcal{S}_r$ denotes retweet.

cial activities, and therefore they may not be well suitable to social media. Let us consider a motivating example. Figure 1 shows the change of the amount of activities of three different types over a 10-hour time window for two queries. If we consider only the total number of tweets, we can see that for both queries there is a burst. However, neither of the two bursts corresponds to a real-world event. The first burst was caused by the broadcast of an advertisement from several Twitter bots, and the second burst was caused by numerous retweets of a status update of a movie star[1]. The detailed explanations of why the two bursts are noisy are also shown in Table 1. On the other hand, interestingly, we can see that not all the activity streams display noisy bursty patterns at the same time. It indicates that we may make use of multiple views of different activities to detect event-related bursts. The intuition is that using multiple types of activities may help learn a better global picture of event-related bursty patterns. Learning may also be more resistant to noisy bursts.

Second, in social media, burst detection is challenged by irregular, unpredictable and spurious noisy bursts. To overcome this challenge, a reasonable assumption is that a burst corresponding to a real event should not fluctuate too much within a relatively short time window. To illustrate it, we present an example in Figure 2, in which we first use a simple threshold method to detect bursts and then analyze the effect of local smoothness. In particular, if the amount of activities at a certain time is above a pre-defined threshold, we set its state to 1, which indicates a bursty state. Otherwise, we set the state to 0. Figure 2(a) shows that for the query "Eclipse," with a threshold of 50, the state sequence for the time window we consider is "0000100000."

[1]The reasons for these bursts were revealed by manually checking the tweets during the corresponding periods.



(a) Query="Eclipse". (b) Query="Nobel".

Figure 2: Analysis of the effect of local smoothness on threshold method. It shows two examples of threshold methods for burst detection in a 10-hour window. The red line denotes the bursty threshold. If the number of activities is above the threshold in one time interval, the state of this time interval is judge as bursty. Detailed descriptions of these cases are shown in Table 1.

Although there is a burst in this sequence, its duration is very short. In fact, this is the first example shown in Table 1, which is a noisy burst. In contrast, in Figure 2(b), the state sequence for the query "Nobel" is "0000011111," in which the longer and smoother burst corresponds to a true event. A good function for evaluating the smoothness of a state sequence should be able to discriminate these cases and model the context of state sequences effectively.

With its unique characteristics and challenges, there is an emergent need to deeply study the problem of event-related burst detection via social media activities. In this paper, we conduct a systematic investigation on this problem. We formulate this problem as burst detection from time series of social media activities. We develop an optimization model to learn bursty patterns based on multiple types of activities. We propose to detect bursts by considering both local state smoothness and correlation across multiple streams. We define a function to

quantitatively measure local smoothness of one single state sequence. We systematically evaluate three types of activities for burst detection on a large Twitter dataset and analyze different properties of these three streams for burst detection.

2 Problem Definition

Before formally introducing our problems, we first define some basic concepts.

Activity: An activity refers to some type of action that users perform when they are interested in some topic or event.

Activity Stream: An activity stream of length N and type m is a sequence of numbers $(n_1^m, n_2^m, ..., n_N^m)$, where each n_i^m denotes the amount of activities of type m that occur during the ith time interval.

Query: A query Q is a sequence of terms $q_1, ..., q_{|Q|}$ which can represent the information needs of users. For example, an example query related to President Obama is "barack obama."

Event-related Burst: Given a query Q, an event-related burst is defined as a period $[t_s, t_e]$ in which some event related with Q takes place, where t_s and t_e are the start timestamp and end timestamp of the event period respectively. During the event period the amount of activities is significantly higher than average.

Based on these definitions, our task is to try to identify event-related bursts via multiple social media activity streams.

3 Identifying Event-related Bursts from Social Media

In this section, we discuss how to identify event-related bursts via social media activities. Formally, given a query Q, we first build M activity streams related with Q on T timestamps: $\{(n_1^m, ..., n_T^m)\}_{m=1}^M$. The definition of *activity* in our methods is very general; it includes various types of social media activities, including textual and non-textual activities, e.g., a click on a shared photo and a link formation between two users.

Given the input, we try to infer a state sequence over these T timestamps: $\boldsymbol{z} = (z_1, ..., z_T)$, where z_i is 1 or 0. 1 indicates a time point within a burst while 0 indicates a non-bursty time point.

3.1 Modeling a Single Activity Stream

3.1.1 Generation function

In probability theory and statistics, the Poisson distribution[2] is a discrete probability distribution that can measure the probability of a given number of "activities" occurring in a fixed time interval. We use the Poisson distribution to study the probability of observing the number of social media activities, and we treat one hour as one time interval in this paper.

Homogeneous Poisson Distribution The generative probability of the ith number in one activity stream of type m is defined as $f(n_i^m, i, z_i^m) = \frac{(\lambda_{z_i^m})^{n_i^m} \exp(-\lambda_{z_i^m})}{n_i^m!}$, where λ_0 is the (normal) expectation of the number of activities in one time interval. If one state is bursty, it would emit activities with a faster rate and result in a larger expectation λ_1. We can set $\lambda_1 = \lambda_0 \times \rho$, where $\rho > 1$.

Heterogeneous Poisson Distribution The two-state machine in (Kleinberg, 2003) used two global references for all the time intervals: one for bursty and the other for non-bursty. In our experiments, we observe temporal patterns of user behaviors, i.e., activities in some hours are significantly more than those in the others. Instead of using fixed global rates λ_0 and λ_1, we try to model temporal patterns of user behaviors by parameterizing $\lambda_{(\cdot)}$ with the time index. By following (Ihler et al., 2006), we use a set of hour-specific rates $\{\lambda_{1,h}\}_{h=1}^{24}$ and $\{\lambda_{0,h}\}_{h=1}^{24}$.[3] Given a time index h, we set $\lambda_{0,h}$ to be the expectation of the number of activities in hth time interval every day, then we have $\lambda_{1,h} = \lambda_{0,h} \times \rho$. In this paper, ρ is empirally set as 1.5.

3.1.2 Smoothness of a State Sequence

For burst detection, the major aim is to identify steady and meaningful bursts and to discard transient and spurious bursts. Given a state sequence $z_1 z_2 ... z_T$, to quantitatively measure the smoothness and compactness of it, we introduce some measures.

One simple method is to count the number of change in the state sequence. Formally, we use the following formula:

$$g_1(\boldsymbol{z}) = T - \sum_{i=1}^{T-1} \mathcal{I}(z_i \neq z_{i+1}), \quad (1)$$

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

[3] We can also make the rates both day-specific and hour-specific, i.e., $\{\lambda_{(\cdot),d,h}\}_{h\in\{1,...,24\},d\in\{1,...,7\}}$.

where T is length of the state sequence and $\mathcal{I}(\cdot)$ is an indicator function which returns 1 only if the statement is true. Let us take the state sequence "0000100000" (shown in Figure 2(a)) as an example to see how g_1 works. State changes $0_{\text{pos}=4} \rightarrow 1_{\text{pos}=5}$ and $1_{\text{pos}=5} \rightarrow 0_{\text{pos}=6}$ each incur a cost of 1, therefore $g_1(0000100000) = 10 - 2 = 8$. Similarly, we can get $g_1(0000000000) = 10$. There is a cost difference between these two sequences, i.e., $\Delta_{g_1} = 2$. Kleinberg (2003) uses state transition probabilities to model the smoothness of state sequences. With simple derivations, we can show that Kleinberg's model essentially also uses a cost function that is linear in terms of the number of state changes in a sequence, and therefore similar to g_1.

In social media, very short noisy bursts like "0000100000" are very frequent. To discard such noises, we may multiply g_1 by a big cost factor to punish short-term fluctuations. However, it is not sensitive to the state context[4] and may affect the detection of meaningful bursts. For example, state change $0_{\text{pos}=4} \rightarrow 1_{\text{pos}=5}$ in "0000111100" would receive the same cost as that of $0_{\text{pos}=4} \rightarrow 1_{\text{pos}=5}$ in "0000100000" although the later is more like a noise.

To better measure the smoothness of a state sequence , we propose a novel context-sensitive function, which sums the square of the length of the maximum subsequences in which all states are the same. Formally, we have

$$g_2(z_1, z_2, ..., z_T) = \sum_{s_i < e_i} (e_i - s_i + 1)^2, \qquad (2)$$

where s_i and e_i are the start index and end index of the ith subsequence respectively. To define "maximum", we have the constraints $z_{s_i} = z_{s_i+1} = ... = z_{e_i}, z_{s_i-1} \neq z_{s_i}, z_{e_i} \neq z_{e_i+1}$. For example, $g_2(0000000000) = 10^2 = 100$, $g_2(0000100000) = 4^2 + 1^2 + 5^2 = 42$, we can see that $\Delta_{g_2} = 100 - 42 = 58$, which is significantly larger than $\Delta_{g_1}(= 2)$. g_2 rewards the continunity of state sequences while punish the fluctuating changes, and it is context-sensitive. State change $0_{\text{pos}=4} \rightarrow 1_{\text{pos}=5}$ in "0000111100" receives a cost of 4,[5] which is much smaller than that of $0_{\text{pos}=4} \rightarrow 1_{\text{pos}=5}$ in "0000100000". g_2 is also sensitive to the position of state changes, e.g., $g_2(0000100000) \neq g_2(0100000000)$.

3.2 Burst Detection from a Single Activity Stream

Given an activity stream $(n_1^m, ..., n_T^m)$, we would like to infer a state sequence over these T timestamps, i.e., to find out the most possible state sequence $\boldsymbol{z} = (z_1^m, ..., z_T^m)$ based on the data, where $z_i^m = 1$ or 0. We formulate this problem as an optimization problem. The cost of a state sequence includes two parts: generation of activities and smoothness of the state sequence. The objective function is to find a state sequence which incurs the minimum cost. Formally, we define the total cost function as

$$Cost(\boldsymbol{z}) = \underbrace{-\sum_{i=1}^{T} \log f(n_i^m, i, z_i^m)}_{\text{generating cost}} + \underbrace{\Big(- \Phi(z_1^m, ..., z_T^m) \times \gamma_1 \Big)}_{\text{smoothness cost}}, \qquad (3)$$

where $\gamma_1 > 0$ is a scaling factor which balance these two parts. $\Phi(\cdot)$ function is the smoothness function, and we can set it as either $g_1(\cdot)$ or $g_2(\cdot)$.

To seek the optimal state sequence, we can minimize Equation 3. However, exact inference is hard due to the exponential search space. Instead of examining the smoothness of the whole state sequence, we propose to measure the smoothness of all the L-length subsequences, so called "local smoothness". The assumption is that the states in a relatively short time window should not change too much. The new objective function is defined as

$$\begin{aligned} Cost(\boldsymbol{z}) \quad = \quad & -\sum_{i=1}^{T} \log f(n_i^m, i, z_i^m) \\ & -\Big(\sum_{i \geq L} \Phi(z_i^m, ..., z_{i+L-1}^m) \Big) \times \gamma_1. \end{aligned} \qquad (4)$$

The objective function in Equation 4 can be solved efficiently by a dynamic programming algorithm shown in Algorithm 1. The time complexity of this algorithm is $\mathcal{O}(T \cdot 2^L)$. Note that the methods we present in Equation 4 and Algorithm 1 are quite general. They are independent of the concrete forms of $f(\cdot)$ and $\Phi(\cdot)$, which leaves room for flexible adaptation or extension in specific tasks. In previous methods (Kleinberg, 2003), L is often fixed as

[4]Here *context* refers to the window of hidden state sequences.

[5]Indeed, g_2 is not designed for a single state change but for the overall smoothness patterns, so we choose a referring sequence generated by making the corresponding state negative to compute the cost, i.e., $|g_2(00000\underline{1}1110) - g_2(00000\underline{0}1110)| = 4$.

2. Indeed, as shown in Figure 2, in some cases, we may need a longer window to infer the global patterns. In our model, L can be tuned based on real datasets. We can seek a trade-off between efficiency and length of context windows.

Algorithm 1: Dynamic Programming for Equation 4.

$d[i][s][z_i...z_{i-L+1}]$ denotes the minimum cost of the first i timestamps with the state subsequence: $z_i...z_{i-L+1}$ and $z_i = s$;
set $d[0][\cdot][\cdot] = 0$;
set $c_1[i] = \log f(n_i^m, i, z_i^m)$;
set $c_2[i] = \Phi(z_i, ..., z_{i-L+1})$;
b, b': previous and current state window are represented as L-bit binary numbers;
for $i = 1$ **to** T **do**
  for $s = 0$ **to** 1 **do**
    for $b = 0$ **to** $2^L - 1$ **do**
      $b' = (b << 1|s)\&(1 << L - 1)$;
      $d[i][s][b'] \longleftarrow \min(d[i][s][b'], d[i-1][s][b] + c_1[i] + c_2[i])$;
    end
  end
end

3.3 Correlating Multiple Activity Streams

In this section, we discuss how to correlate multiple activity streams to learn a global bursty patterns. The hidden state sequences corresponding to these activity streams are not fully independent. An external event may intricate surges in multiple activity streams simultaneously.

We propose to correlate multiple activity streams in an optimization model. The idea is that activity streams related with one query might be dependent, i.e., the states of multiple activity streams on the same timestamp tend to be the same[6]; if not, it would incur a cost. To implement this idea, we develop an optimization model. For convenience, we call the states of each activity stream as "local states" while the overall states learnt from multiple activity streams as "global states".

The idea is that although various activity streams are different in the scale of frequencies, they tend to share similar trend patterns. We incorporate the correlation between local states on the same timestamp.

[6]In our experiments, we compute the cross correlation between different streams with a lag factor δ, we find the cross correlation achieves maximum consistantly when $\delta = 0$.

Formally, we have

$$Cost(\boldsymbol{Z}) = \sum_{m=1}^{M} \left\{ -\sum_{i=1}^{T} \log f(n_i^m, i, z_i^m) - \sum_{i \geq L} \Phi(z_i^m, ..., z_{i+L-1}^m) \times \gamma_1 \right\} + \sum_{i=1}^{T} \sum_{m_1, m_2} \mathcal{I}(z_i^{m_1} \neq z_i^{m_2}) \times \gamma_2, \quad (5)$$

where $\mathcal{I}(\cdot)$ is indicator function, and γ_2 is the cost when a pair of states are different across multiple streams on the same timestamp.

The objective function in Equation 5 can be solved by a dynamic programming algorithm presented in Algorithm 2. The time complexity of this algorithm is $\mathcal{O}(T \cdot 2^{M \cdot L + M})$. Generally, L can be set as one small value, e.g., L =2 to 6, and we can select just a few representative activity streams, i.e., M =2 to 6. In this case, the algorithm can be efficient.

Algorithm 2: Dynamic Programming for Equation 5.

$d[i][z_i^1...z_{i-L+1}^1; ...; z_i^M...z_{i-L+1}^M]$ denotes the minimum cost of the first i timestamps with the local state subsequence $z_i^m...z_{i-L+1}^m$ in the mth stream;
set $d[0][...] = 0$;
$b^l, b^{l'}$: previous and current state windows represented as $M \cdot L$-bit binary numbers;
$c[i, b^l, b^{l'}]$ denotes all the cost in the tth timestamp;
for $i = 1$ **to** T **do**
  for $b_l = 0$ **to** $2^{M \cdot L} - 1$ **do**
    deriving current local state sequences $b^{l'}$ from b^l;
    $d[i][b_l'] \longleftarrow \min(d[i][b^{l'}], d[i-1][b^l] + c[i, b^l, b^{l'}])$;
  end
end

Given M types of activity streams, we can get M (local) state sequences $\{(z_1^m, ..., z_T^m)\}_{m=1}^M$. The next question is how to learn a global state sequence $(z_1^G, ..., z_T^G)$ based on local state sequences. Here we give a few options:

CONJUNCT: we set a global state z_i as bursty if all local states are bursty, i.e., $z_i^G = \cap_{m=1}^M z_i^m$.

DISJUNCT: we set a global state z_i as bursty if one of the local states is bursty, i.e., $z_i^G = \cup_{m=1}^M z_i^m$.

BELIEF: we set a global state z_i as the most confident local state, i.e., $z_i^G = \text{argmax}_m \text{belief}(z_i^m)$. The belief$(\cdot)$ function can be defined as the ratio between generating costs from states z_i^m and $1 - z_i^m$: $\text{belief}(z_i^m) = \frac{f(n_i^m, i, z_i^m)}{f(n_i^m, i, 1 - z_i^m)}$.

Table 2: Basic statistics of our golden test collection.

# of queries	17
Aver. # of event-related bursts per query	19
Min. bursty interval	3 hours
Max. bursty interval	163 hours
Aver. bursty interval	17.8 hours

L2G: we treat the states of one local stream as the global states.

4 Experiments

4.1 Construction of Test Collection

We test our algorithms on a large Twitter dataset, which contains about 200 million tweets and ranges from July, 2009 to December 2009. We manually constructed a list of 17 queries that have high volumes of relevant tweets during this period. These queries have a very broad coverage of topics. Example queries are "Barack Obama", "Apple", "Earthquake", "F1" and "Nobel Prize". For each query, we invite two senior graduate students to manually identify their golden bursty intervals, and each bursty interval is represented as a pair of timestamps in terms of hours. Specifically, to generate the golden standard, given a query, the judges first manually generate a candidate list of external events[7]; then for each event, they look into the tweets within the corresponding period and check whether there is a surge on the frequency of tweets. If so, the judges further determine the start timepoint and end timepoint of it. If there is a conflict, a third judge will make the final decision. We used Cohen's kappa coefficient to measure the agreement of between the first two judges, which turned out to be 0.67, indicating a good level of agreement[8]. We present basic statistics of the test collection in Table 2.

4.2 Evaluation Metrics

Before introducing our evaluation metrics, we first define the Bursty Interval Overlap Ratio (BIOR)

$$\text{BIOR}(f, \mathcal{X}) = \frac{\sum_{f' \in \mathcal{X}} \Delta l(f, f')}{L(f)},$$

f is a bursty interval, $\Delta l(f, f')$ is the length of overlap between f' and f, $L(f)$ is the length of bursty period of f. $\mathcal{X}$ is a set of bursty intervals, BIOR measures the proportion of the timestamps in f which are covered by one of bursty intervals in $\mathcal{X}$. We use BIOR to measure partial match of intervals, because a system may not return all the exact bursty intervals[9]. We show some examples of BIOR in Figure 3.

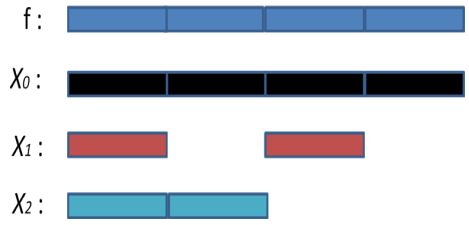


Figure 3: Examples to illustrate BIOR. X_0, X_1 and X_2 are three sets of bursty intervals. X_0 and X_2 consist of one interval, and X_1 consists of two intervals. BIOR(f, X_0)=1, BIOR(f, X_1)=0.5 and BIOR(f, X_2)=0.5.

We use modified Precision, Recall and F as basic measures. Given one query, P, R and F can be defined as follows

$$R = \frac{\sum_{f \in \mathcal{B}} \mathcal{I}\left(\frac{1}{|\mathcal{M}_f|}\text{BIOR}(f, \mathcal{M}) > 0.5\right)}{|\mathcal{B}|},$$

$$P = \frac{1}{|\mathcal{M}|} \sum_{f' \in \mathcal{M}} (\text{BIOR}(f', \mathcal{B})),$$

$$F = \frac{2 \times P \times R}{P + R},$$

where $\mathcal{M}$ is the set of bursty intervals identified by one candidate method, $\mathcal{B}$ is the set of bursty intervals in golden standards, and $\mathcal{M}_f$ is the set of intervals which overlap with f in $\mathcal{M}$. We incorporate the factor $\frac{1}{|\mathcal{M}_f|}$ in Recall to penalize the incontinuous coverage of the golden interval, and we also require that the overlap ratio with penalized factor is higher than a threshold of 0.5. Given two sets of bursty intervals which have the same value of BIOR, we prefer the one with fewer intervals. In Figure 3, we can easily derive X_1 and X_2 have the same value

[7] We refer to some gold news resources, e.g., Google News and Yahoo! News.

[8] http://en.wikipedia.org/wiki/Cohen's_kappa

[9] A simple evaluation method is that we label each one hour time slot as being part of a burst or not and compare with the gold standard. However, in our experiments, we find that some methods tend to break one meaningful burst into small parts and easier to be affected by small fluctuations although they may have a good coverage of bursty points. This is why we adopt a different evaluation approach.

Table 3: Average cross-correlation between different streams.

	$\mathcal{S}_t$	$\mathcal{S}_r$	$\mathcal{S}_u$
$\mathcal{S}_t$	1	0.830235	0.851514
$\mathcal{S}_r$	0.830235	1	0.59905
$\mathcal{S}_u$	0.851514	0.59905	1

of BIOR, when computing Recall, we prefer X_2 to X_1 since X_2 consists of only one complete interval while X_1 consists of two inconsecutive intervals. $\mathcal{I}(\cdot)$ is an indicator function which returns 1 only if the statement if true. In our experiments, we use the average of R, P and F over all test queries.

4.3 Experiment Setup

Selecting activity streams

We consider three types of activity streams in Twitter: 1) posting a tweet, denoted as $\mathcal{S}_t$; 2) forwarding a tweet (retweet), denoted as $\mathcal{S}_r$; 3) posting a URL-embedded tweet, denoted as $\mathcal{S}_u$. It is natural to test the performance of $\mathcal{S}_t$ in discovering bursty patterns, while $\mathcal{S}_u$ and $\mathcal{S}_r$ measure the influence of external events on users in Twitter in two different aspects. $\mathcal{S}_r$: An important convention in Twitter is the "retweeting" mechanism, through which users can actively spread the news or related information; $\mathcal{S}_u$: Another characteristic of Twitter is that the length of tweets is limited to 140 characters, which constrains the capacity of information. Users often embed a URL link in the tweets to help others know more about the corresponding information.

We compute the average cross correlation between different activity streams for these 17 queries in our test collection, and we summarize the results in Table 3. We can see that both $\mathcal{S}_r$ and $\mathcal{S}_u$ have a high correlation with $\mathcal{S}_t$, and $\mathcal{S}_r$ has a relatively low correlation with $\mathcal{S}_u$. [10]

Methods for comparisons

$S_{(\cdot)}$: using Equation 4 and considers a single activity stream, namely S_t, S_u and S_r.

$MBurst_{(\cdot)}$: using Equation 5 and considers multiple activity streams.

To compare our methods with previous methods, we adopt the following baselines:

StateMachine: This is the method proposed in (Kleinberg, 2003). We use heterogeneous Poisson function as generating functions instead of binomial function C_k^n because sometimes it is difficult to get the exact total number n in social media.

Threshold: If we find that the count in one time interval is higher than a predefined threshold, it is treated as a burst. The threshold is set as 1.5 times of the average number.

PeakFinding: This is the method proposed in (Marcus et al., 2011), which aims to automatically discover peaks from tweets.

Binomial: This is the method proposed in (Fung et al., 2007a), which uses a cumulative binomial distribution with a base probability estimated by removing abnormal frequencies.

As for multiple-stream burst detection, to the best of our knowledge, the only existing work is proposed by (Yao et al., 2010), which is supervised and requires a considerable amount of training time, so we do not compare our work with it. We compare our method with the following heuristic baselines:

SimpleConjunct: we first find the optimal state sequences for each single activity stream. We then derive a global state sequence by taking the conjunction of all local states.

SimpleDisjunct: we first find the optimal state sequences for each single activity stream, and then we derive a global state sequence by take the disjunction of all local states.

Another possible baseline is that we first merge all the activities, then apply the single-stream algorithm. However, in our data set, we find that the number of activities in S_t is significantly larger than that of the two types. S_t dominantly determines the final performance, so we do not incorporate it here as a comparison.

4.4 Experimental Results

Preliminary results on a single stream

We first examine the performance of our proposed method on a single stream. Note that, our method in Equation 4 has two merits: 1) the length of local window can be tuned on different datasets; 2) a novel state smoothness function is adopted.

We set the Φ function in Equation 4 respectively as g_1 and g_2, and apply our proposed methods to three streams $(\mathcal{S}_t, \mathcal{S}_r, \mathcal{S}_u)$ mentioned above. Note that, when $L = 2$ and $\Phi = g_1$, our method becomes the algorithm in (Kleinberg, 2003). We tune the parameter γ_1 in Equation 4 from 2 to 20 with a step of 2. We record the best F performance and compute

[10] We also consider the frequencies of unique users by hours, however, we find it has a extremely high correlation coefficient with $\mathcal{S}_t$, about 0.99, so we do not incorporate it.

the corresponding standard deviation. In Table 5, we can observe that 1) streams $\mathcal{S}_t$ and $\mathcal{S}_r$ perform better than $\mathcal{S}_u$; 2) the length of local window significantly affects the performance; 3) g_2 is much better than g_1 in our proposed burst detection algorithm; 4) generally speaking, a longer window size ($L = 3, 4$) performs better than the most common used size 2 in (Kleinberg, 2003).

We can see that our proposed method is more effective than the other baselines. The major reason is that none of these methods consider state smoothness in a systematic way. In our preliminary experiments, we find that these baselines usually output a lot of bursts, most of which are broken meaningful bursts. To overcome this, baseline method StateMachine ($g_1 + L = 2$) requires larger ρ and γ_1, which may discard relatively small meaningful bursts; while our proposed single stream method ($g_2 + L = 3, 4$) tends to identify steady and consecutive bursts through the help of longer context window and context sensitive smoothness function g_2, it is more suitable to be applied to social media for burst detection.

Compared with the other baselines, (Kleinberg, 2003) is still one good and robust baseline since it models the state smoothness partially. These preliminary findings indicate that state smoothness is very important for burst detection, and the length of state context window will affect the performance significantly.

To get a deep analysis of the performance of different streams, we set up three classes, and each class corresponds to a single stream. Since for each query, we can obtain multiple results in different activity streams, we further categorize the 17 annotated queries to the stream which leads to the optimal performance on that query. Interestingly, we can see: 1) the *url* stream gives better performance on queries about big companies because users in Twitter usually talk about the release of new products or important evolutionary news via url-embedded tweets; 2) the *retweet* stream gives better performance on queries which correspond to unexpected or significant events, e.g., diasters. It is consistent with our intuitions that users in Twitter do actively spread such information. Combining previous analysis of Table 5, overall we find the *retweet* stream is more capable to identify bursts which correspond to significant events.

Table 4: Categorization of 17 queries according to the optimal performance.

Streams	Queries
url	*Apple*,*Microsoft*,*Nokia*, climate
retweet	*bomb*,*crash*,*earthquake*,*typhoon*, F1,Google,Olympics
all_tweet	Amazon, eclipse, Lakers, NASA, Nobel Prize, Barack Obama

Table 5: Performance (average F) on a single stream. "⋆⋆" indicates that the improvement our proposed single-stream method$_{g_2,L=4}$ over all the other baselines is accepted at the confidence level of 0.95, i.e., StateMachine, PeakingFinding, Binomial and Threshold.

Φ	L	$\mathcal{S}_t$	$\mathcal{S}_r$	$\mathcal{S}_u$
g_2	4	0.545/0.015	0.543/0.037	0.451/0.036
	3	0.536/0.013	**0.549**⋆⋆/0.019	0.464/0.025
	2	0.468/0.055	0.542/0.071	0.455/0.045
g_1	4	0.513/0.059	0.546/0.058	0.465/0.047
	3	0.469/0.055	0.542/0.071	0.455/0.045
	2	0.396/0.043	0.489/0.074	0.374/0.035
StateMachine		0.396	0.489	0.374
PeakFinding		0.410	0.356	0.302
Binomial		0.315	0.420	0.341
Threshold		0.195	0.181	0.175

Preliminary results on multiple streams

After examining the basic results on a single stream, we continue to evaluate the performance of our proposed models on multiple activity streams. For $MBurst$ in Equation 5, we have three parameters to set, namely L, γ_1 and γ_2. We do a grid search for both γ_1 and γ_2 from 1 to 12 with a step of 1, and we also examine the performance when $L = 2, 3, 4$. We can see that $MBurst$ has four candidate methods to derive global states from local states; for *L2G*, we use the states of $\mathcal{S}_t$ as the final states, and we empirically find that it performs best compared with the other two streams in *L2G*.

Recall that our proposed single-stream method is better than all the other single-stream baselines, so here single-best denotes our method in Equation 4 ($\Phi = g_2, L = 4$) on $\mathcal{S}_r$. For *SimpleConjunct* and *SimpleDisjunct*, we first find the optimal state sequences for each single activity stream using our proposed method in Equation 4 ($\Phi = g_2, L = 4$), and then we derive a global state sequence by take the conjunction or disjunction of all local states respectively.

Besides the best performance, we further compute the average of the top 10 results of each method by tuning parameters to check the average perfor-

Table 6: Performance (average F) on multiple streams. "$\star$" indicates that the improvement our proposed multiple-stream method over our proposed single-stream method at the confidence level of 0.9 in terms of average performance.

Methods	best	average
single-best ($g_2 + \mathcal{S}_r$)	0.549	0.526
SimpleConjunct	0.548	-
SimpleDisjunct	0.465	-
$MBurst$+**CONJUNCT**$_{r,t,u}$	0.555	0.548
$MBurst$+**DISJUNCT**$_{r,t,u}$	**0.576**	**0.570***
$MBurst$+**BELIEF**$_{r,t,u}$	0.568	0.561
$MBurst$+**L2G**$_{r,t,u}(t)$	0.574	0.567
$MBurst$+**L2G**$_{r,t,u}(r)$	0.560	0.558

mance. The average performance can show the stability of models in some degree. If one model outputs the maximum in a very limited set of parameters, it may not work well in real data, especially in social media.

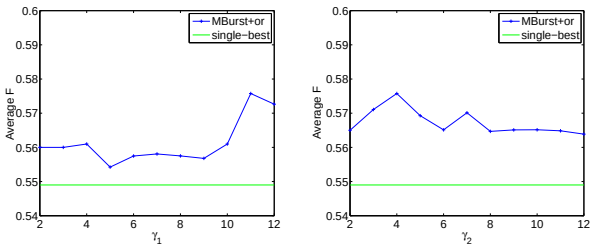
In Table 6, we can see $MBurst$+**DISJUNCT**$_{r,t,u}$ gives the best performance. $MBurst$ performs consistently better than *single-best* which is a very strong single-stream method, especially for average performance. $MBurst$+**DISJUNCT**$_{r,t,u}$ has an improvement of average performance over *single-best* by 8.4%. And simply combining three different streams may hit results (*SimpleConjunct* and *SimpleDisjunct*). It indicates that $MBurst$ is more stable and shows a higher performance.

For different methods to derive global bursty patterns, we can see that $MBurst$+**DISJUNCT** performs best while $MBurst$+**CONJUNCT** performs worst. Interestingly, however, *SimpleConjunct* is better than *SimpleDisjunct*, the major reason is that $MBurst$ performs a local-state correlation of multiple activity streams to correct possible noisy fluctuations from single streams before the conjunction or disjunction of local states. After such correlation, the performance of each activity stream should improve. To see this, we present the optimal results of a single stream without/with local-state correlation in Table 7. Local-state correlation significantly boosts the performance of a single stream. Indeed, we find that the step of local-state correlation is more important for our multiple stream algorithm than the step of how to derive global states based on local states.

We test our $MBurst$ algorithm with the setting: $T = 4416$, $L = 4$ and $M = 3$, and for all the test queries, our algorithm can respond in 2 seconds [11], which is efficient to be deployed in social media.

Table 7: Comparison between the optimal results of a single stream with/without local-state correlation.

	all_retweet	retweet	url
without	0.536	0.549	0.464
with	0.574	0.560	0.547



(a) $\gamma_2 = 4$, varying γ_1. (b) $\gamma_1 = 11$, varying γ_2.

Figure 4: Parameter sensitivity of $MBurst$ + **DISJUNCT**.

Parameter sensitivity

We have shown the performance of different parameter settings for single stream algorithm in Table 5. Next, we check parameter sensitivity in $MBurst$. In our experiments, we find a longer local window ($L = 3, 4$) is better than $L = 2$, so we first set $L = 4$, then we select parameter settings of $\gamma_2 = 4$ and $\gamma_1 = 11$, which give best performance for $MBurst$+**DISJUNCT**. We vary one with the other fixed to see how one single parameter affects the performance. The results are shown in Figure 4, and we can see $MBurst$+**DISJUNCT** is consistently better than single-best.

5 Related Work

Our work is related to burst detection from text streams. Pioneered by the automaton model proposed in (Kleinberg, 2003), many techniques have been proposed for burst detection such as the χ^2-test based method (Swan and Allan, 2000), the parameter-free method (Fung et al., 2005) and moving average method (Vlachos et al., 2004). Our work is related to the applications of these burst detection algorithms for event detection (He et al., 2007; Fung et al., 2007b; Shan et al., 2012; Zhao et al., 2012).

[11] All experiments are tested in a Mac PC, 2.4GHz Intel Core 2 Duo.

Some recent work try to identify hot trends (Mathioudakis and Koudas, 2010; Zubiaga et al., 2011; Budak et al., 2011; Naaman et al., 2011) or make use of the burstiness (Sakaki et al., 2010; Aramki et al., 2011; Marcus et al., 2011) in social media. However, few of these methods consider modeling the local smoothness of one state sequence in a systematic way and often use a fixed window length of 2.

Little work considers making use of different types of social media activities for burst detection. (Yao et al., 2010; Kotov et al., 2011; Wang et al., 2007; Wang et al., 2009) conducted some preliminary studies of mining correlated bursty patterns from multiple sources. However, they either highly relies on high-quality training datasets or require a considerable amount of training time. Online social activities are dynamic, with a large number of new items generated continuously. In such a dynamic setting, burst detection algorithms should effectively collect evidence, efficiently adjust prediction models and respond to the users as social media activities evolve. Therefore it is not suitable to deploy such algorithms in social media.

Our work is also similar to studies which aim to mine and leverage knowledge from social media (Mathioudakis et al., 2010; Ruiz et al., 2012; Morales et al., 2012). We share the common point with these studies that we try to utilize the underlying rich knowledge in social media, while our focus of this work is quite different from theirs, i.e., to identify event-related bursts.

Another line of related research is Twitter related studies (Kwak et al., 2010; Sakaki et al., 2010). Our proposed methods can provide event-related bursts for downstream applications.

6 Conclusion

In this paper, we propose to identify event-related bursts via social media activities. We propose one optimization model to correlate multiple activity streams to learn the bursty patterns. To better measure local smoothness of the state sequence, we propose a novel state cost function. We test our methods in a large Twitter dataset. The experiment results show that our methods are both effective and efficient. Our work can provide a preliminary understanding of the correlation between the happenings of events and the degree of online social media activities.

Finally, we present a few promising directions which may potentially improve or enrich current work.

1) **Variable-length context.** In this paper, L is a pre-determined parameter which controls the size of context window. It cannot be modified when the algorithm runs. A large L will significantly increases the algorithm complexity, and we may not need a large L for all the states in a Markov chain. This problem can be addressed by using the variable-length hidden Markov model (Wang et al., 2006), which is able to learn the "minimum" context length for accurately determining each state.

2) **Incorporation of more useful features.** Our current model mainly considers temporal variations of streaming data and searches the surge patterns existing in it. In some cases, simple frequency information may not be capable to identify all the meaningful bursts. It can be potentially useful to leverage up more features to help filter out noisy bursts, e.g., semantic information (Zhao et al., 2010).

3) **Modeling multi-modality data.** We have examined our multi-stream algorithm by using three different activity streams. These streams are textual-based. It will be interesting to check our algorithm in multi-modality data streams. E.g., in Facebook, we may collect a stream consisting of the daily frequencies of photo sharing and another stream consisting of the daily frequencies of text status updates.

4) **Evaluation of the identified bursts.** In most of previous work, they seldom construct a gold standard for quantitative test, instead they qualitatively evaluate their methods. In our work, we invite human judges to generate the gold standard. It is time-consuming, and the bias from human judges cannot be completely eliminated although more judges can be invited. A possible evaluation method is to examine the identified bursts in downstream applications, e.g., event detection.

Acknowledgement

This work is partially supported by NSFC Grant 61073082, 60933004 and 70903008. Xin Zhao is supported by Google PhD Fellowship (China). We thank the insightful comments from Junjie Yao and the anonymous reviewers.

References

Eiji Aramki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: Detecting influenza epidemics using twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1576, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Structural trend analysis for online social networks. *Proc. VLDB Endow.*, 4, July.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu. 2005. Parameter free bursty events detection in text streams. In *VLDB*.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Huan Liu, and Philip S. Yu. 2007a. Time-dependent event hierarchy construction. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Huan Liu, and Philip S. Yu. 2007b. Time-dependent event hierarchy construction. In *SIGKDD*.

Qi He, Kuiyu Chang, and Ee-Peng Lim. 2007. Analyzing feature trajectories for event detection. In *SIGIR*.

Alexander Ihler, Jon Hutchins, and Padhraic Smyth. 2006. Adaptive event detection with time-varying poisson processes. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, pages 207–216, New York, NY, USA. ACM.

J. Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*.

Alexander Kotov, ChengXiang Zhai, and Richard Sproat. 2011. Mining named entities with temporally correlated bursts from multilingual web news streams. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM, pages 237–246.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600.

Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. 2011. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11.

Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 1155–1158.

Michael Mathioudakis, Nick Koudas, and Peter Marbach. 2010. Early online identification of attention gathering items in social media. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 301–310, New York, NY, USA. ACM.

Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. 2012. From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *WSDM*, pages 153–162.

Mor Naaman, Hila Becker, and Luis Gravano. 2011. Hip and trendy: Characterizing emerging trends on twitter. *JASIST*, 62(5):902–918.

Eduardo J. Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. 2012. Correlating financial time series with micro-blogging activity. pages 513–522.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. WWW, pages 851–860, New York, NY, USA. ACM.

Dongdong Shan, Wayne Xin Zhao, Rishan Chen, Shu Baihan, Hongfei Yan, and Xiaoming Li. 2012. Eventsearch: A system for event discovery and retrieval on multi-type historical data. In *KDD'12*, Demostration.

Russell Swan and James Allan. 2000. Automatic generation of overview timelines. In *SIGIR*.

Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. 2004. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*.

Yi Wang, Lizhu Zhou, Jianhua Feng, Jianyong Wang, and Zhi-Qiang Liu. 2006. Mining complex time-series data by learning markovian models. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM, pages 1136–1140, Washington, DC, USA. IEEE Computer Society.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Xiang Wang, Kai Zhang, Xiaoming Jin, and Dou Shen. 2009. Mining common topics from multiple asynchronous text streams. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM, pages 192–201.

Junjie Yao, Bin Cui, Yuxin Huang, and Xin Jin. 2010. Temporal and social context based burst detection from folksonomies. In *AAAI*.

Wayne Xin Zhao, Jing Jiang, Jing He, Dongdong Shan, Hongfei Yan, and Xiaoming Li. 2010. Context modeling for ranking and tagging bursty features in text

streams. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10.

Wayne Xin Zhao, Rishan Chen, Kai Fan, Hongfei Yan, and Xiaoming Li. 2012. A novel burst-based text representation model for scalable event detection. In *ACL'12*.

Arkaitz Zubiaga, Damiano Spina, Víctor Fresno, and Raquel Martínez. 2011. Classifying trending topics: a typology of conversation triggers on twitter. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM.