# Twevent: Segment-based Event Detection from Tweets

Chenliang Li, Aixin Sun, Anwitaman Datta
School of Computer Engineering, Nanyang Technological University, Singapore
{lich0020,axsun,anwitaman}@ntu.edu.sg

## ABSTRACT

Event detection from tweets is an important task to understand the current events/topics attracting a large number of common users. However, the unique characteristics of tweets (*e.g.,* short and noisy content, diverse and fast changing topics, and large data volume) make event detection a challenging task. Most existing techniques proposed for well written documents (*e.g.,* news articles) cannot be directly adopted. In this paper, we propose a segment-based event detection system for tweets, called *Twevent*. *Twevent* first detects bursty tweet segments as event segments and then clusters the event segments into events considering both their frequency distribution and content similarity. More specifically, each tweet is split into non-overlapping segments (*i.e.,* phrases possibly refer to named entities or semantically meaningful information units). The bursty segments are identified within a fixed time window based on their frequency patterns, and each bursty segment is described by the set of tweets containing the segment published within that time window. The similarity between a pair of bursty segments is computed using their associated tweets. After clustering bursty segments into candidate events, Wikipedia is exploited to identify the realistic events and to derive the most newsworthy segments to describe the identified events. We evaluate *Twevent* and compare it with the state-of-the-art method using 4.3 million tweets published by Singapore-based users in June 2010. In our experiments, *Twevent* outperforms the state-of-the-art method by a large margin in terms of both precision and recall. More importantly, the events detected by *Twevent* can be easily interpreted with little background knowledge because of the newsworthy segments. We also show that *Twevent* is efficient and scalable, leading to a desirable solution for event detection from tweets.

## Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Clustering; Information filtering*

## Keywords

Event detection, Twitter, Microblogging, Tweet segmentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*CIKM'12,* October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.

## 1. INTRODUCTION

Twitter, as a social networking service and microblogging service, has gained great success in recent years. Twitter users not only share and communicate with friends and family, but also to the general public. The unique gene of the latter enables Twitter users to access and contribute "*the latest stories, ideas, opinions, and news*[1]" about what many users find interesting. As a result, Twitter has become one of the top-10 most visited website on the Internet[2]. As of March 2012, there are more than 140 million active users with over 340 million tweets published a day[3].

### 1.1 Motivation

In Twitter, each user becomes an individual news media that not only absorbs/assembles information (such as breaking news), but also publishes/propagates opinions, sentiments, and stories of themselves [9, 20]. The message unit, called a *tweet*, is limited to maximum 140 characters in length. Such a concise unit enables information updates at extremely low-cost and in realtime, making Twitter a timely fresh information resource [3]. Consequently, the intensive interaction between users in realtime enables timely event detection by monitoring tweet updates and many prominent events are timely spotlighted by Twitter users. For example, a record number of tweet updates per second was set within a 30-sec period after the 2010 FIFA World Cup match between Japan and Cameroon on June 14, 2010. Three days later, the record was broken right after the Lakers' victory in the 2010 NBA Finals on June 17, 2010[4]. Other than sports related events, it has been reported that earthquake detection based on Twitter is faster than the detection based on traditional media [19]. Moreover, event detection from tweets would help us gain timely understanding of users' opinion/sentiment with respect to the detected events, making it possible for company/organization to take a fast response to any emerging crisis. Event detection from Twitter stream would also contribute to the study of mass communication by analyzing the types of events general users are mostly interested in [22] as well as the reactions by users at different geographical regions [15].

Event detection from Twitter stream is challenging for at least three reasons: short and noisy content, diverse and fast changing topics, and large data volume. The task of event detection has been intensively studied in the past mostly on formal texts, *e.g.,* news articles, blog posts, or academic papers [4–6, 8]. However, tweets are significantly different from well written texts because of the shortness and informal writing style. According to the principle of

[1] https://twitter.com/about
[2] http://www.alexa.com/siteinfo/twitter.com
[3] http://blog.twitter.com/2012/03/twitter-turns-six.html
[4] http://bits.blogs.nytimes.com/2010/06/18/sports-fans-break-records-on-twitter/

least effort [26], people are used to communicate information with the least context, especially in the situation where a short message with free style is allowed. This makes tweets contain a lot of misspellings and informal abbreviations [17]. Because of the noise and shortness, direct adoption of most existing approaches developed for formal texts (*e.g.,* clustering bursty features with co-occurrence measure [5, 6]) is doomed to fail on Twitter streams.

Tweets cover very diverse topics and about half of the tweets are not event-related according to a study by PearAnalytics [12]. They manually categorized 2,000 tweets into six categories: news (3.6%), spam (3.75%), self-promotion (5.85%), pointless babble (40.55%), conversational (37.55%) and pass-along value (8.7%). The numbers indicate the percentage of the tweets in each category. Based on their analysis, about 50% (*i.e.,* spam, self-promotion, pointless babble) of tweets are not related to events. Similar observations are also made in our pilot study of the tweets data used in our experiments. However, a large number of features would be expected being bursty from tweets of *pointless babble* category. Obviously, none of these bursty features would help in detecting any event, but would mislead the event detection algorithm and also incur unnecessary computational cost. The situation would be further exaggerated with the fast changing topics in tweets. For example, many users would discuss about a football match during the match or within a few hours right after the match but not for a few days.

## 1.2 Overview of Twevent

To address the above challenges, we present *Twevent*, a novel segment-based event detection system for tweets. One novel feature of *Twevent* is to use the notion of *tweet segment* instead of unigram to detect and describe events. A tweet segment is one or more consecutive words (or phrase) in a tweet message. We observe that tweet segments contained in a large number of tweets are likely to be named entities (*e.g.,*Steve Jobs) or some semantically meaningful unit (*e.g.,*Argentina vs Nigeria). Therefore, a tweet segment often contains much more specific information than any of the unigrams contained in the segment. The use of tweet segment instead of unigrams therefore greatly reduces the noise in the event detection process and also makes the event detected much easier to be interpreted. For example, *Twevent* detected an event with the following five segments [south korea, greece, korea vs greece, korea won, korea] on 12 June 2010; the event is self-explanative. Another novel feature of *Twevent* is the utilization of external knowledge base in guiding the event detection process. In the following, we brief the main steps in *Twevent* for event detection from tweets.

Given tweets published in a Twitter stream, *Twevent* firstly segments each individual tweet into a sequence of consecutive phrases (*i.e.,* segments). Then bursty segments are identified by modeling the frequency of a segment as a Gaussian distribution based on predefined fixed time-window (*e.g.,* a day or an hour). To detect events attracting a larger number of users, we also utilize user frequency (or user support) of the tweet segments to identify the event-related bursty segments, called event segments. After that, we apply an efficient clustering algorithm to group event-related segments as candidate events, which requires only a single pass through each pair of event segments. To compute the similarity between a pair of event segments, we consider the frequency distribution and the content of the tweets containing each of the tweet segments published within the time-window. The result of event segment clustering is a set of candidate events detected in that time window. The knowledge encoded in Wikipedia is then harnessed to help us figure out the realistic events detected from the trivial ones and to derive the most representative segments for describing the realistic events. As the result, each event detected by *Twevent* is represented by a ranking list of segments including many named entities for easy interpretation.

*Twevent* holds several features to address the challenges of event detection from tweets. Tweet segmentation employed in *Twevent* identifies informative phrases which reduces noise in further processing. The use of user frequency in bursty event segment extraction makes *Twevent* robust to the negative impact of the tweets of *Spam* and *Self-Promotion*. The external knowledge base offers *Twevent* the ability to resist the adverse impact of diverse and dynamic topics of tweets, such as tweets of *Pointless Babble*, and derive interpretable event descriptions. Lastly, *Twevent* is efficient and scalable by utilizing only the frequency of segments for bursty segment extraction and non-iterative clustering algorithm.

We evaluated *Twevent* with more than 4.3 million tweets published by Singapore-based users over a one month period. In our experiments, *Twevent* achieves much better performance compared to the state-of-the-art method [21] in terms of both precision and recall. More specifically, *Twevent* achieves a precision of 86.1%, and a recall of 75 distinct events detected from the one-month data. Our experimental results also demonstrate the effectiveness of using tweet segments compared to the same detection process using unigrams. To illustrate that the events detected by *Twevent* often contain named entities or convey concise information, we list the most newsworthy segments detected by *Twevent* in Table 2 as part of the experimental results.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 describes *Twevent* and its components in detail. Section 4 presents the experimental results. We conclude this paper in Section 5.

# 2. RELATED WORK

Event detection has a long history, which can be traced back to the Topic Detection and Tracking (TDT) project[5], which is to detect and track events from news stream. Two main approaches have been studied in the literature: *document-pivot* and *feature-pivot* approaches. The former aims to cluster documents related to the same events and then extract event-based features from the document clusters [1, 2, 23–25]. The latter aims to firstly identify the representative features of the hidden events from the stream, which are assumed to have bursty frequency patterns along time. Then events are detected by clustering these representative features [8]. Because the proposed *Twevent* is a feature-pivot method, in our literature survey, we therefore mainly focus on *feature-pivot* methods for event detection from formal texts.

## 2.1 Event Detection from Formal Texts

Kleinberg [8] proposed to detect events by analyzing frequency patterns along time. An infinite-state automation is used to model the changes of word frequency, and the state transitions are considered as events. Fung *et al.* [5] proposed to identify bursty features as representatives for the events hidden in text stream. The frequency of each feature (*i.e.,* unigram word) is modeled with a binomial distribution. The bursty feature extraction is then based on the perspective of statistics. The events are then detected by maximizing the co-occurrences among documents and the consistence of the frequency distributions for all bursty features within an event. The timestamp for an event is calculated based on the bursty periods of the bursty features related to that event. The authors further presented an event-based search framework in [4] to retrieve groups of documents such that the documents in each group are about the same event. In their result, the events are organized

[5] http://projects.ldc.upenn.edu/TDT/

in a time-based hierarchy. In this event-based framework, a set of related bursty features with similar frequency distributions are retrieved firstly. Then, documents related to the bursty features are extracted and clustered into a hierarchy of events. Instead of using frequency directly, He *et al.* [6] proposed to use *Discrete Fourier Transformation* (DFT) to extract bursty features. They build a signal for each feature using *document frequency - inverse document frequency* ($df \times idf$) scheme along time domain. Then, DFT transforms the signal in time domain to frequency domain, *i.e.,* a spike in frequency domain indicates a corresponding high frequency signal source. Similar to [5], they group bursty features into events by considering both features' co-occurrence and their distributions in time domain. To estimate the timestamp of the events, they model a $df \times idf$ signal with a Gaussian mixture.

Unlike formal texts that are formally written and published in moderate rate, tweets are short, informally written, and published at an enormous amount. Thus, bursty feature extraction solely based on statistics would result in a huge number of bursty features, particulary when unigram feature representation is used. Similarly, the application of DFT would be dread and prohibited. Further, co-occurrence measures used in [5,6] may not work well in the context of twitter due to sparsity.

## 2.2 Event Detection from Tweets

Recently, event detection on twitter stream becomes a hot research topic. Michael and Nick [11] presented a trend detection system over twitter stream. They firstly identify the bursty terms based on queueing theory. Then bursty terms are grouped into the events based on their co-occurrences. For a detected trend, PCA, SVD and entity extraction techniques are then applied to derive contextual information for the trend description. Petrović *et al.* [13] tracked events on twitter stream by applying locality sensitive hashing (LSH). LSH is applied to each tweet to measure the similarity to existing tweets. The tweets similar to each other are grouped as events. Swit and Tsuyoshi [14] proposed an approach for breaking news detection and tracking by clustering the similar tweets together. The approach only focuses on the tweets with a specific hashtag #breakingnews. The similarity between two tweets of breaking news is measured by using a variant of $tf \cdot idf$ scheme where the named entities detected by a Named Entity Recognizer (NER) are further boosted. Popescu *et al.* [16] proposed a method for entity-based event detection on twitter streams. A set of tweets containing the predefined target entity are processed and machine learning techniques are used to predict whether the tweets constitute an event regarding the entity. Very recently, Li *et al.* [18] proposed to detect crime and disaster related Events (CDE) from tweets. Conventional text mining techniques are applied to extract the meta information (*e.g.,* geo-location names, temporal phrase, and keywords) for event interpretation. To summarize, most existing approaches for detecting events from tweets are applicable to certain types of tweets (*e.g.,* having a specific hashtag, containing a predefined entity, or related to crime and disaster). The other solutions including [11] and [13] involve complicated processing and lead to heavy computational cost.

The most related work to ours, is the approach proposed by Weng and Lee, named *EDCoW* [21]. There are three steps in their approach. Firstly *wavelet transformation* and *auto correlation* are applied to measure the bursty energy of each word. The words with outstanding high energies are retained as event features. Then they measure the similarity between each pair of event features by using *cross correlation*. At last, modularity-based graph partitioning is used to detect the events, each of which contains a set of words with high *cross correlation*. However, several issues get in the way of the practical application for their approach. *Wavelet transformation* and *auto correlation* for each word of the twitter stream would require a huge amount of computation, making it not a scalable choice. Moreover, utilizing only *cross correlation* for similarity measure would lead to the resulted event consisting of several distinct events which happened at the same period by coincidence (*e.g.,* two football matches hold at the same time during FIFA 2010 World Cup). Thirdly, the detected events with unigram features are difficult for human interpretation. In *Twevent*, we segment each tweet into possible semantic phrases, making the detected events easy to interpret. During the detection process, we do not employ computational costly Wavelet transformation and auto correlation for tweet segments. Instead, only the tweet frequency and user frequency are needed for bursty tweet segment detection. To distinguish events that happened at the same period, *Twevent* computes content similarity for a pair of tweet segments. Each tweet segment is described by the content of the tweets containing the segment. Although pair-wise similarity computation is computational costly, it is only applied to a relatively small set of bursty tweet segments detected within one time window.

# 3. TWEVENT

In this section, we present a feature-pivot event detection framework. Illustrated in Figure 1, our framework consists of three main components: *tweet segmentation*, *event segment detection*, and *event segment clustering*. After receiving a tweet from a Tweet stream, tweet segmentation component splits the tweet into non-overlapping segments. A tweet segment can be either a unigram or multi-gram (*e.g.,*[mtv movie awards], [steve jobs]), and each segment may or may not represent a semantic unit. The resultant tweet segments obtained from a tweet, together with the content and timestamp of the tweet, are indexed in the segment index. The event segment detection component detects abnormal bursty segments by considering tweets frequency distribution and user frequency of the segments. The event segments about the same event are then grouped together to form the event by the event clustering component. In the rest of this section, we describe each component in detail following the order of their usage in our framework.

## 3.1 Tweet Segmentation

The notion of *tweet segment* was firstly proposed in our recent work [10] for named entity recognition, not related to event detection. In the following, we brief the techniques for tweet segmentation.

Given a tweet $d \in \mathcal{T}$, the problem of tweet segmentation is to split $d$ into $m$ non-overlapping and consecutive segments, $d = \langle s_1 s_2 ... s_m \rangle$, where a segment $s_i$ is either a word (or unigram) or a phrase (or multi-gram). We formulate tweet segmentation problem as an optimization problem with the following objective function, where $C$ is the function measures the *stickiness* of a segment or a tweet.

$$\arg\max_{s_1,\ldots,s_m} C(d) = \sum_{i=1}^{m} C(s_i), \tag{1}$$

A high *stickiness* score of segment $s$ indicates that further splitting segment $s$ would break the correct word *collocation*. In other words, segment $s$ cannot be further split at any internal position if it has a high stickiness score. We define *stickiness* function by using the generalized *Symmetric Conditional Probability* (SCP) for $n$-grams with $n \geq 2$, supported by statistical information derived from Microsoft Web N-Gram service[6] and Wikipedia.

[6] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx

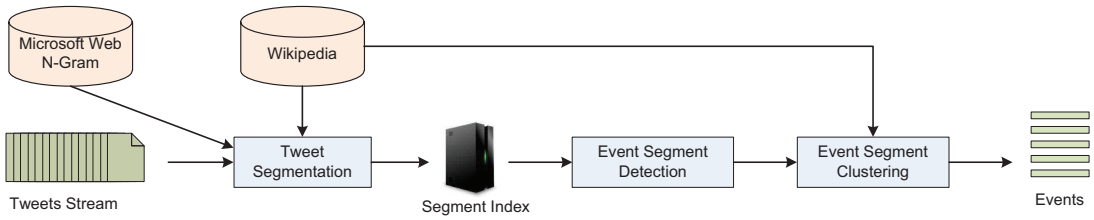



**Figure 1: Segment-based Event Detection System Architecture**

SCP is defined to measure the "cohesiveness" of a segment $s = \langle w_1 \ldots w_n \rangle$ $(n > 1)$ by considering all possible binary segmentations, as shown in the following equation, where $\Pr(\cdot)$ denotes the prior probability derived from Microsoft Web N-Gram service.

$$SCP(s) = \log \frac{\Pr(s)^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} \Pr(w_1 \ldots w_i)\Pr(w_{i+1} \ldots w_n)} \quad (2)$$

In this equation, the logarithm value is taken to avoid underflow. Note that, $SCP(s) = 2\log Pr(w)$ if segment $s$ is of unit length (*i.e.,* $|s| = 1$ or $n = 1$).

Based on $SCP(s)$, we define the stickiness score of segment $s$ by considering the high quality semantic resources in Wikipedia. That is, segments which frequently appear as anchor texts in Wikipedia are further favored. The *stickiness* function is then defined as:

$$C(s) = \mathcal{L}(s) \cdot e^{Q(s)} \cdot \mathcal{S}(SCP(s)) \quad (3)$$

$$\mathcal{L}(s) = \begin{cases} \frac{|s|-1}{|s|}, \text{ for } |s| > 1 \\ 1, \text{ for } |s| = 1 \end{cases} \quad (4)$$

where $Q(s)$ is the probability that $s$ appears as the anchor text in the Wikipedia articles that contain $s$, and $\mathcal{S}(\cdot)$ is the sigmoid function. The function $\mathcal{L}$ defined in Equation 4 is used to give moderate preference for longer segments. With this stickiness function, the tweet segmentation defined in Equation 1 can be finished efficiently in linear time with dynamic programming.

## 3.2 Event Segment Detection

One salient characteristic of emerging events in text streams is that there is a significant coverage of topics related to an event within a certain time period. Accordingly, given a collection of segments of the tweets published within a fixed time window, bursty segments in terms of frequency would be potentially related to some hot events talked and shared by Twitter users. However, considering the dynamic nature and the large volume of tweets published everyday, efficiently detecting bursty segments is non-trivial.

Let $N_t$ denote the number of tweets published within time-window $t$ from Twitter stream, $f_{s,t}$ be the number of tweets containing $s$ published within $t$, *i.e.,* the tweet frequency of segment $s$ in time-window $t$. The probability of observing frequency $f_{s,t}$ of segment $s$ in $t$ can be modeled by a binomial distribution [5].

$$P(f_{s,t}) = \binom{N_t}{f_{s,t}} p_s^{f_{s,t}} (1-p_s)^{N_t - f_{s,t}} \quad (5)$$

where $p_s$ is the expected probability of tweets that contain segment $s$ in a random time window. Given that $N_t$ is very large in the case of Twitter stream, it is reasonable to approximate $P(f_{s,t})$ with Gaussian distribution:

$$P(f_{s,t}) \sim \mathcal{N}(N_t p_s, N_t p_s(1-p_s)). \quad (6)$$

Thus, given segment $s$, the expected number of tweets containing $s$ would be $E[s|t] = N_t p_s$. The more the additional tweets containing $s$ with respect to $E[s|t]$, the more bursty the segment is. On the other hand, segment $s$ with frequency $f_{s,t} <= E[s|t]$ is considered as a non-bursty segment and will not be considered for further processing. Hence, we define bursty segment as follows.

*Definition 1.* **[Bursty Segment]** A segment $s$ is a bursty segment in time window $t$ if its tweet frequency $f_{s,t} > E[s|t]$.

Next we transfer the frequency of a bursty segment into range of $(0, 1]$ indicating its ***bursty probability***.

We consider a bursty segment $s$ to be extremely bursty and assign $P_b(s,t) = 1$ if its tweet frequency $f_{s,t} \geq E[s|t] + 2\sigma[s|t]$, where $\sigma[s|t] = \sqrt{N_t p_s(1-p_s)}$ is the standard deviation based on Equation 6. For a bursty segment whose tweet frequency $f_{s,t}$ falls within the range $(E[s|t], E[s|t] + 2\sigma[s|t])$, we use the following equation to compute its bursty probability.

$$P_b(s,t) = \mathcal{S}\left(10 \times \frac{f_{s,t} - (E[s|t] + \sigma[s|t])}{\sigma[s|t]}\right) \quad (7)$$

where $\mathcal{S}(\cdot)$ is the sigmoid function, and a constant 10 is introduced in the equation because the sigmoid function $\mathcal{S}(x)$ smooths reasonable well for $x$ in the range of $[-10, 10]$.

With the above statistical method, we are able to detect the bursty segments and assign each a bursty probability. However, Twitter is significantly different from most text streams (*e.g.,* news stream and blog stream) that have been extensively studied in the literature for bursty feature/event detection, because of its informal writing style and topic diversity. Therefore, a large number of tweet segments would be detected to be bursty segments. A simple statistics in our study shows that the number of distinct bursty segments is about 75% of the number of distinct tweets in a randomly chosen time window. Among the bursty segments detected, many contain misspelling words and informal abbreviations. These noisy bursty segments would not only incur unnecessary computational cost but also hurt the event detection accuracy in the further processing. We therefore source for the wisdom of the crowds to filter the bursty segments.

Instead of solely relying on the tweet frequency of a segment, we believe that a bursty segment has a higher chance to be related to an event if there are more users post tweets containing the segment. Hence, we define ***user frequency*** $u_{s,t}$ of a segment $s$, which is the number of users who post tweets containing $s$ during the time period $t$.

With the two factors, bursty probability and user frequency, the most simple approach to detect the event-relatedness of a bursty segment is to take the product of the two factors. However, this simple approach would make the top-ranked segments dominated by the ones used by most users, such as "i'm", "i'll", and "guys". To some extent, bursty segments with higher user frequencies are correlated with some events. However, considering the limited length of tweets, the bursty segments with higher user frequencies may not be semantically meaningful and are often ambiguous. For instance, "nigeria", "argentina" and "argentina vs nigeria" are all related to a single event: *a 2010 world cup match between nigeria and ar-*

*gentian*. However, the bursty segment "argentina vs nigeria" has a relatively much lower user frequency due to the principle of the least effort [26]. In contrast, comparing "argentina vs nigeria" with either "nigeria" or "argentina", the segment "argentina vs nigeria" would convey much more information about the event. Based on this observation, we assign each bursty segment $s$ a weight $w_b(s,t)$ by using a logarithm function.

$$w_b(s,t) = P_b(s,t)\log(u_{s,t}) \tag{8}$$

The above weight scheme would keep the more bursty segments of the higher user frequency being ranked higher and the more bursty segments of the moderate user frequency being ranked relatively higher than the others.

By ranking the bursty segments by their weights $w_b(s,t)$, we then retain the top-$K$ bursty segments as potential event-related segments (or simply event segments) for further processing. The value of $K$ is non-trivial because a small $K$ would result in a very low recall of events detected, and a large $K$ may bring in more noise, leading to much higher computational cost as well as lower precision on the detected events. In practical, the optimal $K$ value depends on the size of the time window, and requires some expertise knowledge (*e.g.,*, users from different regions may be interested in different topics [15]). In this work, we apply a heuristic strategy to filter out the bursty segments by setting $K$ to $\sqrt{N_t}$.

*Definition 2.* **[Event Segment]** A bursty segment $s$ is a potential event-related segment (or simply event segment) in time window $t$ if it is ranked among top-$K$ bursty segments by $w_b(s,t)$ in descending order, where $K = \sqrt{N_t}$.

## 3.3 Event Segment Clustering

Given a set of event segments detected from the previous step, we now cluster them into groups, each of which corresponds to a *possible* realistic event. Some event segments that cannot be clustered into groups are considered noise or non-event-related. These non-event-related segments are dropped from further processing.

### 3.3.1 Event Segment Similarity

Accordingly, we need to derive a similarity measure for each pair of event segments. Various similarity measures have been used in the past to cluster bursty features detected in formal texts, mainly based on co-occurrences of bursty features [5, 6]. However, similarity measure based on co-occurrence would not work well on tweets because they are much shorter in number of words compared to formal documents. Moreover, the topics in tweets are extremely dynamic and fast changing. Considering these two factors, we propose to measure similarity between two event segments by the content of their associated tweets and their temporal frequency patterns.

For each time window $t$, we further divide the time period evenly into $M$ sub-time-window: $t = \langle t_1 \ldots t_M \rangle$. The tweet frequency of an event segment $s$ in sub-window $t_m$ is denoted by $f_t(s,m)$. Let $T_t(s,m)$ be the set of tweets that each contains segment $s$ and is published within sub-window $t_m$. We define the similarity between a pair of segments $s_a$ and $s_b$ within time window $t$ as follows:

$$sim_t(s_a,s_b) = \sum_{m=1}^{M} w_t(s_a,m)w_t(s_b,m)sim\left(T_t(s_a,m), T_t(s_b,m)\right) \tag{9}$$

where $sim(T_1,T_2)$ measures the similarity between two sets of tweets $T_1$ and $T_2$, and $w_t(s,m)$ weighs the importance of sub-window $t_m$ to segment $s$. To compute $sim(T_1,T_2)$, we concatenate all tweets in $T_1$ (resp. $T_2$) to form a pseudo document, and use cosine similarity with $tf \cdot idf$ scheme. The importance of sub-window $t_m$ to segment $s$ is the normalized frequency distribution over $M$ sub-windows:

$$w_t(s,m) = \frac{f_t(s,m)}{\sum_{m'=1}^{M} f_t(s,m')} \tag{10}$$

Equation 9 illustrates that two event segments are similar if they have both similar tweet content and consistent frequency patterns along the time sub-windows. Either dissimilar tweet content or inconsistent frequency patterns leads to low similarity. More specifically, dissimilar tweet content suggests that two event segments refer to two distinct events. Inconsistent frequency pattern may suggest that the two event segments refer to two similar events but happened at different time points (*e.g.,* two football matches at the same day).

To be shown in our experiments (Section 4.3) content similarity is necessary to distinguish segments of different events having very similar tweet frequency distributions. Note that, because of the specific information conveyed by tweet segment, we believe that the content similarity of using all tweets containing the tweet segment is more meaningful than that using unigram. For example, the tweets containing segment steve jobs will be very different from the tweets containing either steve or the tweets containing jobs only.

### 3.3.2 Clustering by $k$-Nearest Neighbor Graph

Given the similarity measure in Equation 9, clustering event segments into possible events become straightforward and many existing clustering algorithms can be directly applied. We apply an variant of Jarvis-Patrick algorithm [7] for event segment clustering.

Given a graph of objects with edges indicating the similarity between any two objects, Jarvis-Patrick clustering algorithm partitions the graph by measuring the number of common neighbors among the $k$-nearest neighbors of the two objects. The partitioning involves two parameters: $k$ and $\ell$. Two objects are put into the same cluster if: 1) they are in each others' $k$-nearest neighbors, and 2) they share at least $\ell$ common nearest neighbors among the $k$-nearest neighbors. Note that, Jarvis-Patrick requires a single scan of all pairs of objects for clustering, which offers great scalability for Twitter stream-based event detection.

Considering the unique properties of short length and informal writing style of tweets, two event segments referring to the same event may not share a large number of common $k$-nearest neighbors to each other. Nevertheless, an event segment referring to a realistic event would likely appear in another event segment's $k$-nearest neighbors, and vice versa, given that the two event segments referring to the same event. We therefore relax the clustering criterion by considering only the first requirement: *two event segments appearing in each others' k-nearest neighbors are put into the same cluster*. With this relaxation, given a complete graph of event segments, the clustering becomes to retain any edge between two event segments $s_a$ and $s_b$ if and only if they appear in each other's $k$-nearest neighbors. The resultant connected components are considered as ***candidate events***. If an event segment is in isolation and not grouped into any cluster, it is considered not event related and dropped from further processing. The clustering of event segments therefore requires only one parameter $k$ and we set $k = 3$ in our experiments.

### 3.3.3 Candidate Event Filtering

Cambridge Dictionaries Online defines an *event* as "anything that happens, especially something important and unusual"[7]. However, we observe that many candidate events detected through clustering event segments are not realistic events. For instance, the

[7] http://dictionary.cambridge.org/dictionary/british/event?q=event

segments "friday night", "friday", "weekends", "trip" and "enjoy" are returned as a possible event by the above procedures at some Friday (*e.g.,* Jun 18, 2010 covered in our dataset). More detailed human investigation shows that the tweets of this candidate event are from people who were talking about the plan or schedule for the coming weekend. Apparently, this kind of events can not be considered as realistic events. This calls for a mechanism to evaluate the "important and unusual" aspect of a candidate event obtained from event segment clustering.

We observe that many events involve well-known entities (*e.g.,* person names, locations, festivals) and many of these entities are documented in Wikipedia. Recall that each segment is produced in Section 3.1 with the preference towards Wikipedia entities (see Equation 3). We therefore again utilize Wikipedia to approximately evaluate "important and unusual" aspect of a candidate event. More specifically, we define *newsworthiness* measures for event segment and candidate event respectively.

*Definition 3.* [**Segment Newsworthiness**] The newsworthiness $\mu(s)$ of a segment $s$ is

$$\mu(s) = \max_{\ell \in s} e^{Q(\ell)} - 1$$

where $\ell$ is any sub-phrase of $s$, and $Q(\ell)$ is the prior probability that $\ell$ appears as anchor text in Wikipedia articles that contain $\ell$.

The exponential function is used in the equation since it is an increasing function with an increasing first derivative in the range of $[0, 1]$. That is, a segment with a larger $Q(\ell)$ would gain a relatively higher *newsworthiness* value. Next, we define the *newsworthiness* of a candidate event as follows.

*Definition 4.* [**Event Newsworthiness**] The newsworthiness $\mu(e)$ of an event $e$ containing a set of event segments $e_s = \{s\}$ is

$$\mu(e) = \frac{\sum_{s \in e_s} \mu(s)}{|e_s|} \cdot \frac{\sum_{g \in E_e} sim(g)}{|e_s|}$$

where $E_e$ is a set of edges that are retained during applying Jarvis-Patrick clustering, and $sim(g)$ is the similarity of edge $g$ which is calculated by using Equation 9.

Observe that *newsworthiness* of a candidate event considers both the newsworthiness of its member event segments (*i.e.,* the first component) and the topology of the connected component formed by its member event segments (*i.e.,* the second component). The latter is equivalent to measure the density of the connected component in the clustering result. Therefore, a candidate event receives a high newsworthiness score if some phrases in its member segments are commonly used as anchor text in Wikipedia (indicating well known entities) and the member segments are well connected with strong cohesive topology.

We observe in our experiments, most top-ranked candidate events by newsworthiness are likely related to realistic events. On the other hand, noisy events likely have much lower *newsworthiness* scores. That is, the distribution of newsworthiness scores has a positive skewness. Let $\mu_x$ be the highest newsworthiness score among all candidate events detected in a given time-window. Based on the above observation, we consider a candidate event $e$ to be a realistic event if the ratio between $\mu_x$ and $\mu(e)$ is smaller than a threshold $\tau$, *i.e.,* $\frac{\mu_x}{\mu(e)} < \tau$, otherwise noise. Naturally, a lower threshold results in better precision of the detected events but poorer recall, and vice versa. We investigate the impact of $\tau$ empirically in Section 4.

After filtering away noisy events, we represent each detected event with its member event segments sorted by *newsworthiness* scores. The top-ranked segments are used to describe the event. In this work, the top-5 segments are used to describe the event.

## 3.4 Discussion

The efficiency of *Twevent* is a non-trivial factor from a practical perspective. Recall that *Twevent* contains three main components: tweet segmentation, event segment detection, and event segment clustering, shown in Figure 1. We next discuss the computational cost for each component.

The running time of tweet segmentation is linear to the length of a tweet (in number of the words). As segmentation of one tweet is independent of segmentation of other tweets, parallel computing techniques can be easily utilized in this component. More importantly, tweet segmentation can be considered as a part of preprocess because all segments are stored in an index for further processing.

Event segment detection only requires one scan of segments' tweet frequency and user frequency. The time complexity is linear to the number of segments in each time window.

Most computation time is consumed by calculating the similarity between two event segments, and event segment clustering. However, the pair-wised event segment similarity is computed for a relatively small set (*e.g.,* $K = \sqrt{N_i}$) of event segments where $N_i$ is the number of tweets published within a time window. That is, the time complexity is $O(K^2)$. The Jarvis-Patrick clustering algorithm used for event segment clustering requires one scan of all pairs of event segments within a time window. Given the relative small number of detected events in each time window, the running time for the candidate event filtering is negligible.

We conducted our experiments on a workstation with a 2.40GHz Xeon quad-core CPU and 24GB of RAM. Without considering the time taken for tweet segmentation[8], *Twevent* takes about 18 seconds to detect events from average $143K$ tweets published in one day (*i.e.,* the time window).

# 4. EXPERIMENTS

In this section, we report our extensive experiments on evaluating *Twevent*. We show that *Twevent* outperforms the state-of-the-art approach with both better precision and recall. We show that newsworthy segments make the detected events much easier to be interpreted by users. Further, we evaluate the usefulness of the notion of tweet segment against unigram, the effect of the parameters in *Twevent*.

## 4.1 Dataset and Experimental Setting

**Wikipedia Data.** The Wikipedia data used in tweet segmentation (Section 3.1) and *newsworthiness* measure (Section 3.3.3) are based on the Wikipedia dump released on 30 Jan, 2010. It contains $3,246,821$ articles and $266,625,017$ hyperlinks. In total, there are $4,342,732$ distinct entities appeared as anchor texts in the Wikipedia dump.

**Twitter Stream.** A collection of tweets published by Singapore-based users (based on the location specified in user profile) in June 2010 is used to simulate a Twitter stream. This dataset was built by Weng and Lee for evaluating the *EDCoW* event detection method in [21]. There are a total of $4,331,937$ tweets published by $19,256$ unique users in the dataset. A number of realistic events happened in the data collection period, such as FIFA World Cup 2010, WWDC 2010, and MTV Movie Awards 2010.

Figure 2(a) shows the average number of tweets published within each hour of a day. Most tweets are published within 6AM to 6PM, with relatively more tweets published in the afternoon.

**Parameter Setting.** There are several parameters that could affect

[8]In our experiments, the speed of tweet segmentation relies on Microsoft Web N-Gram Web service.

Table 1: Detection results of *Twevent*, *Twevent*$_u$, and *EDCoW*.

| Method | No. events detected | Precision | Recall | DERate |
|---|---|---|---|---|
| *EDCoW* | 21 | 76.2% | 13 | 23.1% |
| *Twevent* | 101 | **86.1%** | 75 | **16.0%** |
| *Twevent*$_u$ | 146 | 75.3% | **78** | 41.0% |

the performance of *Twevent*. The size of the time-window $t$ and the number of sub-time-windows $M$ are the two basic parameters for *Twevent*. In our evaluation, we fix $t$ to be a day and set $M = 12$. That is, each sub-time-window is 2 hours.

Recall that we retain only top $K = \sqrt{N_t}$ bursty segments as event segments, where $N_t$ is the number of tweets published in the time window $t$; For Jarvis-Patrick clustering, we set $k = 3$ for the number of nearest neighbors in the graph; To distinguish realistic events from noise, we set threshold on the ratio of newsworthiness $\tau$. In our experiments, we observe that parameter $\tau$ affects the event detection accuracy of *Twevent* more significantly than the other two parameters $K$ and $k$. We therefore evaluate the impact of varying $\tau$ and fix $K = \sqrt{N_t}$ and $k = 3$ throughout our evaluation.

**Evaluation Metric.** The dataset does not come with ground truth labels on all realistic events within the data collection period. Because it is infeasible to manually label the over 4 million tweets in the dataset, we choose to manually evaluate the detected events returned by *Twevent*. We use *precision* and *recall* to evaluate the accuracy of the events detected.

We follow the definition of **Precision** used in [21], which is defined as the fraction of the detected events that are related to a realistic event. However, recall was not defined in [21] because of the lack of the ground truth labels in the dataset. In our paper, we choose to report **Recall** as *the number of distinct realistic events detected* from the dataset on daily basis. Note that, if two detected events are both related to the same realistic event within the same time window (*i.e.,* one day), then both are considered correct in terms of precision, but only *one* realistic event is considered in counting recall. Due to this reason, we also define *Duplicate Event Rate* (or simply **DERate**) to denote the percentage of events that have been duplicately detected among all realistic events detected.

## 4.2 Statistics on Tweet Segmentation

We first report statistics on the tweet segments returned by the tweet segmentation component. After removal of stop-words and words with non-English characters, there are 662, 088 distinct words or unigrams retained in the data. Tweet segmentation is then applied to each tweet. A total of 1, 275, 809 distinct segments are obtained after segmentation.

Observe from Figure 2(b), the tweet frequency of a segment follows a power-law distribution. On average, each segment is contained in 19 tweets. In Figure 2(c), we report the distribution of unigram and multi-gram segments. The figure shows that about 50% of the segments are 2-grams; segments with more than 3 grams are very rare. We observe that 2-gram segments cover a large proportional of named entities, such as "lady gaga" and "justin bieber", or location name like "orchard road". Moreover, many 3-grams segments are very informative, like "mtv movie awards" and "penalty shoot out". More examples are given in Table 2.

## 4.3 Event Detection Results

We compare *Twevent* with two methods: *EDCoW* and *Twevent*$_u$. The latter is a variant of *Twevent* without using tweet segment but using unigram in the event detection process, with the same parameter settings as *Twevent* except the setting for threshold $\tau$. In this set

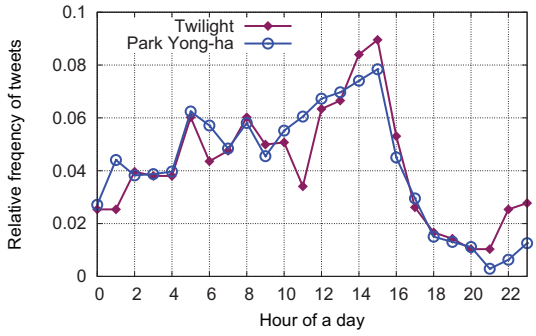



**Figure 3: Comparison of frequency distributions for the two events: Twilight and Park Yong-ha**

of experiments, we set $\tau = 4$ for *Twevent* and $\tau = 3$ for *Twevent*$_u$. The impact of varying $\tau$ will be reported in Section 4.4.

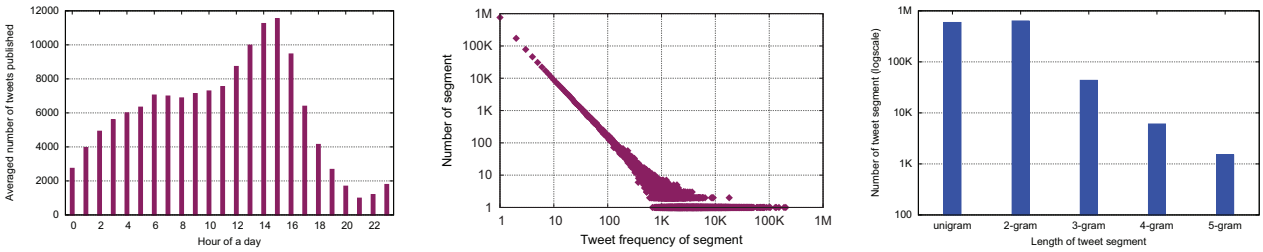
Recall that we measure the segment's newsworthiness using Definition 3. Based on this definition, a unigram is likely to have zero newsworthiness score. Thus, we change the newsworthiness definition for unigram to be $\mu(w) = e^{Q(w)}$ for a unigram $w$. The modified definition strongly favors informative unigrams, leading to better representations for the detected events. Next, we report the event detection accuracy of the three methods.

**Event detection accuracy**. Table 1 reports the number of events detected, the precision and recall, of the three methods respectively. The results of *EDCoW* are reproduced from [21][9]. Shown in the table, our proposed method *Twevent* yields the best precision of 86.1% which is significantly larger than the precisions achieved by *EDCoW* and *Twevent*$_u$. Observe that our method *Twevent* detects 101 events with a recall of 75 realistic events. On the same dataset, *EDCoW* detects 21 events in total with 13 realistic events. *Twevent*$_u$ yields a slighter worse precision than *EDCoW* (75.3% vs 76.2%) but detects the largest number of realistic events. In terms of DERate, *Twevent* achieves the lowest rate despite that our method detects much more events than *EDCoW* (101 vs 21). On the other hand, we observe that *Twevent*$_u$ delivers the worst DERate, more than double of *Twevent* (41% vs 16.0%). That is, the unigrams about the same event are clustered into two or more events. Because a tweet segment usually conveys very specific information, the tweets containing the tweet segment are all about the same topic (*e.g.,* the event). Two tweet segments about the same event are therefore have higher chance to be clustered together.

From the list of events detected, we observe that an event is re-detected mainly because users discuss the event from different perspectives, or one event is a sub-event of another. We use the two events $e_{22}$ and $e_{20}$ detected on 12 Jun 2012 as an example for illustration. Listed in Table 2, $e_{22}$, detected with segments [usa, england, eng, vs], refers to the football match between England and USA in 2010 World Cup; $e_{20}$ with [steven gerrard, captain, score, scored, gerrard] refers to the caption of England, Steve Gerrard, scored a goal in this match. Because both events refer to the same match, we count them as one realistic event in our recall.

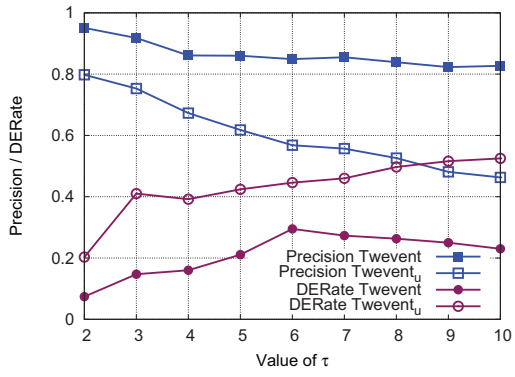
**Event interpretation**. We argue that the notion of *tweet segment* not only benefits better precision in event detection, but also makes the detected events much easier to be interpreted. In above discussion, we show that person name [steven gerrard] is detected as an event segment in the result. We now give more examples by listing all the events detected by *Twevent* between June 07 to

[9] All events detected by *EDCoW* are reported in Table 1, Page 407 in [21].

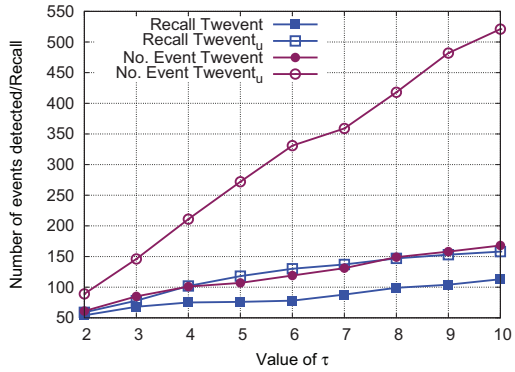



(a) Tweet volume against hour of day (b) Segment tweet frequency distribution (c) Segment length distribution

**Figure 2: Statistics on tweets and tweet segments**




(a) Precision and DERate




(b) Recall and number of detected event

**Figure 4: *Twevent* and *Twevent*$_u$ against different $\tau$ values**

June 12, 2010[10]. This 6-day period is chosen because it covers a wide range of events that happened in June, 2010, including Apple WWDC 2010, MTV Movie Awards 2010, and FIFA World Cup 2010, among others.

From Table 2, we make two observations. First, many event segments are multi-gram segments such as various types of named entities [steve jobs], [mtv movie awards], [katy perry], and [karate kid], and segments conveying concrete information like [argentina vs nigeria] and [season finale]. These segments make the events much easier to be interpreted than some unigram keywords. For comparison, we also list the keywords of all the 8 events detected by *EDCoW*, reproduced from [21], and the keywords of all the 40 events detected by *Twevent*$_u$ during the 6-day period in Tables 3 and 4, respectively. We can see that the keywords detected by *EDCoW* is relatively hard to interpret than the two methods *Twevent* and *Twevent*$_u$. Compare the latter two methods, we argue that *Twevent* detects more semantically meaningful keywords/phrases than *Twevent*$_u$. For instance, we use the first two events detected by both methods as example. Although both methods detect similar keywords for the WWDC 2010 event, [steve jobs, imovie, wwdc, iphone, wifi] is more easy to interpret than [wwdc, keynote, live, jobs, steve]. Similarly, *Twevent* detects [mtv movie awards, mtv, new moon, twilight, awards] while *Twevent*$_u$ outputs [mtv, moon, twilight, movie, awards]; the phrase [mtv movie awards] makes the event much more easy to interpret. Second, the detected events by *Twevent* cover a wide range of events, such as Korea music bands, Apple WWDC 2010, MTV Movie Awards 2010, release of music videos and movies, and football matches of World Cup 2010. That is, *Twevent* does not favor certain types of events than others. Among the 22 events listed in Table 2, only 1 event has no corresponding real-life events leading to a precision of 95.5% in this 6-day period.

[10]Due to page list, we do not list all the 101 events detected by *Twevent* for the whole month of June 2010. The full list is available at `http://www.cais.ntu.edu.sg/~lichenliang/twevent/eventlist.txt`.

**Table 3: Events detected by *EDCoW* in June 07 – June 12, 2010 (reproduced from [21], following their event id.)**

| Day | $e_{ID}$ | Event keywords |
|---|---|---|
| 7 | $e_7$. | kobe, kristen |
| | $e_8$. | #iphone4, ios4, iphone |
| 8 | $e_9$. | reformat, hamilton |
| | $e_{10}$. | avocado, commence, ongoing |
| 9 | $e_{11}$. | #failwhale, twitter |
| 10 | $e_{12}$. | vuvuzela, soccer |
| 11 | $e_{13}$. | #svk, #svn |
| 12 | $e_{14}$. | #kor, greec, #gre |

**Case study**. We use a case study to illustrate the importance of using content in event detection from tweets. Figure 3 plots the relative frequency of tweets published on June 30, 2010 related to two events. The first event with segments [harry potter, twilight, hours, trailer] is about the movie *The Twilight Saga: Eclipse*, which was released on June 30, 2010. The trailer of the Movie *Harry Potter and the Deathly Hallows Part 1* was shown before *The Twilight Saga: Eclipse*. The second event, with segments [park yong ha, rip, peace, hope, park] refers to the suicide case of Korean actor Park Yong-ha on June 30, 2010. Observe from Figure 3, the two events have very similar tweet frequency distribution over the 24 hours of the day. The tweet frequency of an event is defined as the relative frequency of the tweets published in an hour that contain any event segment of the event. We argue that it is hard to distinguish the two events without using content similarity between the event segments (see Section 3.3).

**Table 2: List of the 22 events detected by *Twevent* during the period of June 07 to June 12, 2010.**

| Day | $e_{ID}$ | [Event Segments]: Event Description |
|---|---|---|
| 7 | $e_1$. | **[steve jobs, imovie, wwdc, iphone, wifi]**: iPhone4 was released during WWDC 2010. |
| | $e_2$. | **[mtv movie awards, mtv, new moon, twilight, robe]**: The movie *The Twilight Saga: New Moon* was the biggest winner in MTV Movie Awards 2010; it took 4 out of 10 "Best" Awards. |
| | $e_3$. | **[yesung, yesung oppa, kyuhyun, oppa, kyu]**: Korean popular band Super Junior's showcase was held on June 6, 2010 at Singapore. Yesung Oppa and Kyuhyun Oppa are members of Super Junior. |
| 8 | $e_4$. | **[lady gaga, music video, gaga, mv, alejandro]**: The music video *Alejandro* by Lady GaGa was premiered officially on June 8, 2010. |
| | $e_5$. | **[ss501, indonesia, ariel, sama, trend]**: No clear corresponding real-life event. |
| | $e_6$. | **[singapore, iphone 4g, iphone 3gs, iphone, coming out]**: Related to event $e_1$. People started to talk about the release date of iPhone 4 in Singapore. |
| 9 | $e_7$. | **[lady gaga, youtube, youtube video, music video, gaga]**: Related to event $e_4$. |
| | $e_8$. | **[twitter, whale, stupid, capacity, over again]**: 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. |
| | $e_9$. | **[ipad, iphone, apple, new]**: Related to event $e_1$. |
| | $e_{10}$. | **[watching glee, glee, season finale, season, channel]**: The season finale of the American TV series *Glee* was broadcasted on June 8, 2010. |
| 10 | $e_{11}$. | **[lady gaga, youtube, youtube video, music video, amber]**: Related to event $e_7$. |
| | $e_{12}$. | **[justin bieber, try, pa, took, each]**: Related to event $e_{15}$. The song *Never Say Never* by Justin Bieber serves as the theme song for the movie *The Karate Kid*, which was released on June 10, 2010 in Singapore. |
| | $e_{13}$. | **[yesung, tweeted]**: Super Junior's Yesung posted a photo about his pet turtles. |
| | $e_{14}$. | **[twitter, whale, stupid, capacity, over]**: Related to event $e_8$. |
| | $e_{15}$ | **[karate kid, watch movie, movie]**: The movie *The Karate Kid* was released on June 10, 2010 in Singapore. |
| 11 | $e_{16}$. | **[uruguay vs france, uruguay, france, vs]**: A match between Uruguay and France in World Cup 2010. |
| | $e_{17}$. | **[south africa, vs mexico, mexico, goal, first goal]**: A match between South Africa and Mexico in World Cup 2010. And the first goal of the 2010 World Cup was scored in the match. |
| 12 | $e_{18}$. | **[arg, argentina, argentina vs nigeria, nigeria, messi]**: A match between Argentina and Nigeria in World Cup 2010. |
| | $e_{19}$. | **[south korea, greece, korea vs greece, korea won, korea]**: A match between South Korea and Greece in World Cup 2010. |
| | $e_{20}$. | **[steven gerrard, captain, gerrard, scores]**: Related to event $e_{22}$. The captain of England, Steve Gerrard scored a goal in the match. |
| | $e_{21}$. | **[ji sung, park, scored, jisung]**: Related to event $e_{19}$. Park Ji-Sung, the caption of South Korea, scored a goal against Greece. |
| | $e_{22}$. | **[usa, england, eng, vs]**: A match between England and USA in World Cup 2010. |

## 4.4 Impact of $\tau$ in Twevent

While event segments are clustered into candidate events, the ratio threshold $\tau$ defines the boundary between the realistic events and the noisy events. We next analyze the effect of $\tau$ value on the performance of *Twevent* and *Twevent*$_u$.

Figure 4(a) plots the precision and DERate of the two methods when changing $\tau$ from 2 to 10. We make the two main observations. First, for both methods, precision degrades along the increase of $\tau$ as more events are considered as realistic events. Nevertheless, the rate of degradation for *Twevent* is much smaller than that of *Twevent*$_u$. Observe that *Twevent* maintains very good precision above 80% even when $\tau = 10$, which is better than the best precision achieved by *Twevent*$_u$ with all $\tau$ values. Second, increasing of $\tau$ leading to increase in DERate. Shown in Figure 4(a), the DERate of *Twevent* is about half of *Twevent*$_u$ for most $\tau$ values and stop increasing when $\tau > 6$. In summary, the use of tweet segmentation in event detection contribute to much higher precision and much lower duplicate event detection rate.

Figure 4(b) reports the number of events detected and the recall values along the change of $\tau$ values for the two methods. For both methods, increase $\tau$ leads to better recall. Observe that, although *Twevent*$_u$ always achieves better recall than *Twevent*, the differences between the recall values do not change much along the increase of $\tau$. However, increment of $\tau$ leads to sharp increase of the number of events detected for *Twevent*$_u$. But due to the poorer precision along the increase of $\tau$, the number of realistic events detected does not increase at the same pace as the number of detected events.

# 5. CONCLUSION

Twitter, as a new type of social media, has experienced an explosive growth in terms of both users and information volume in recent years. The characteristics of tweets propose severe challenges to many tasks including event detection. In this paper, we present a novel event detection system for Twitter stream, called *Twevent*, to tackle the adverse impacts of tweets: short and noisy content, diverse and dynamic topics, and large data volume. One of the key concept in *Twevent* is to use tweet segment instead of unigram for identifying the bursty features and then distinguishing the realistic events from the noisy ones. *Twevent* demonstrates outstanding performance in our experiments: effectiveness, informativeness, and efficiency. As a part of our future work, we will investigate the effectiveness of utilizing more features from tweets (*e.g.,* retweet rate and hashtags) in *Twevent*. Another important task is to investigate the effectiveness of *Twevent* when none of the segments of an event is covered by Wikipedia.

# 6. ACKNOWLEDGEMENT

This work was partially supported by MOE AcRF Tier-1 Grant RG13/10, Singapore. We thank Jianshu Weng from HP Labs Singapore for providing us the dataset used in [21].

**Table 4: Events detected by *Twevent*$_u$ in June 07 – June 12, 2010**

| Day | $e_{ID}$ | Event keywords |
|---|---|---|
| 7 | $e_1$. | wwdc, keynote, live, jobs, steve |
| | $e_2$. | mtv, moon, twilight, movie, awards |
| | $e_3$. | yesung, ryeowook, oppa, hope, welcome |
| | $e_4$. | indonesia, blues, rain, weather, star |
| | $e_5$. | iphone, apple, video, available, os |
| | $e_6$. | singapore, asia, showcase, junior, tickets |
| | $e_7$. | check, website, gain, member, site |
| | $e_8$. | justin, both, pa, took, each |
| | $e_9$. | perry, katy |
| | $e_{10}$. | singtel, samsung, galaxy, cute, aniversary |
| | $e_{11}$. | tumblr, white, black, hair, model |
| 8 | $e_{12}$. | gaga, lady, mv, alejandro |
| | $e_{13}$. | singapore, iphone, apple, july, features |
| | $e_{14}$. | wwdc, keynote, jobs, steve |
| | $e_{15}$. | right, think, one, need, know |
| 9 | $e_{16}$. | twitter, whale, capacity, fail, over |
| | $e_{17}$. | fan, wish, pretty, follow |
| | $e_{18}$. | youtube, glee, season, video, finale |
| | $e_{19}$. | gaga, lady, mv, alejandro |
| | $e_{20}$. | rice, chicken, eat, food, lunch |
| | $e_{21}$. | mtv, asia, wednesday, awards |
| | $e_{22}$. | try, pa, took |
| | $e_{23}$. | home, out, going, i'm |
| 10 | $e_{24}$. | day, work, one, need, know |
| | $e_{25}$. | youtube, gaga, video, lady, liked |
| | $e_{26}$. | twitter, whale, stupid, capacity, fail |
| | $e_{27}$. | radio, station, heard, addicted |
| | $e_{28}$. | justin, try, pa, took |
| | $e_{29}$. | karate, movie, kid, watched |
| | $e_{30}$. | congrats, blue, jay, name, jaywalkers |
| | $e_{31}$. | boss, centre, thursday, suntec, fair |
| | $e_{32}$. | facebook, internet, camera, photo, photos |
| | $e_{33}$. | cup, world, opening |
| 11 | $e_{34}$. | mexico, africa, goal, scored, south |
| | $e_{35}$. | uruguay, france |
| 12 | $e_{36}$. | park, ji, sung, jisung |
| | $e_{37}$. | argentina, nigeria, usa, england, messi |
| | $e_{38}$. | captain, steve, gerrard, scores |
| | $e_{39}$. | greece, korea, koreans, han, min |
| | $e_{40}$. | goalkeeper, robe, green, ball, mistake |

# 7. REFERENCES


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

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

[3] A. Dong, R. Zhang, P. Kolari, J. Bai, F. Diaz, Y. Chang, Z. Zheng, and H. Zha. Time is of the essence: improving recency ranking using twitter data. In *WWW*, pages 331–340, 2010.

[4] G. P. C. Fung, J. X. Yu, H. Liu, and P. S. Yu. Time-dependent event hierarchy construction. In *SIGKDD*, pages 300–309, 2007.

[5] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *VLDB*, pages 181–192, 2005.

[6] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *SIGIR*, pages 207–214, 2007.

[7] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Trans. Comput.*, 22(11):1025–1034, Nov. 1973.

[8] J. Kleinberg. Bursty and hierarchical structure in streams. In *SIGKDD*, pages 91–101, 2002.

[9] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[10] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee. Twiner: Named entity recognition in targeted twitter stream. In *SIGIR*, 2012.

[11] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*, pages 1155–1158, 2010.

[12] PearAnalytics. Twitter study, August 2009. Available online `http://www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf`.

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

[14] S. Phuvipadawat and T. Murata. Breaking news detection and tracking in twitter. In *WI-IAT*, pages 120–123, 2010.

[15] B. Poblete, R. Garcia, M. Mendoza, and A. Jaimes. Do all birds tweet the same?: characterizing twitter around the world. In *CIKM*, pages 1025–1030, 2011.

[16] A.-M. Popescu, M. Pennacchiotti, and D. Paranjpe. Extracting events and event descriptions from twitter. In *WWW*, pages 105–106, 2011.

[17] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *EMNLP*, pages 1524–1534, 2011.

[18] L. Rui, L. Kin, K. Ravi, and C. Kevin. Tedas: a twitter based event detection and analysis system. In *ICDE*, pages 1273–1276, 2012.

[19] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, pages 851–860, 2010.

[20] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: a comparison of microblog search and web search. In *WSDM*, pages 35–44, 2011.

[21] J. Weng and B.-S. Lee. Event detection in twitter. In *ICWSM*, pages 401–408, 2011.

[22] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on twitter. In *WWW*, pages 705–714, 2011.

[23] Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer. Improving text categorization methods for event tracking. In *SIGIR*, pages 65–72, 2000.

[24] Y. Yang, J. G. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, July 1999.

[25] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *SIGIR*, pages 28–36, 1998.

[26] G. K. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Hafner Pub. Co, 1949.