Event Detection in *Twitter*

Jianshu Weng
Services Platform Lab, HP Labs Singapore
jianshu.weng@hp.com

Bu-Sung Lee
Services Platform Lab, HP Labs Singapore and
School of Computer Engineering, Nanyang Technological University
francis.lee@hp.com

Abstract

Twitter, as a form of social media, is fast emerging in recent years. Users are using *Twitter* to report real-life events. This paper focuses on detecting those events by analyzing the text stream in *Twitter*. Although *event detection* has long been a research topic, the characteristics of *Twitter* make it a non-trivial task. Tweets reporting such events are usually overwhelmed by high flood of meaningless "babbles". Moreover, event detection algorithm needs to be scalable given the sheer amount of tweets. This paper attempts to tackle these challenges with *EDCoW* (*E*vent *D*etection with *C*lustering of *W*avelet-based Signals). *EDCoW* builds signals for individual words by applying wavelet analysis on the frequency-based raw signals of the words. It then filters away the trivial words by looking at their corresponding signal auto-correlations. The remaining words are then clustered to form events with a modularity-based graph partitioning technique. Experimental results show promising result of EDCoW.

Introduction

Microblogging, as a form of social media, is fast emerging in recent years. One of the most representative examples is *Twitter*, which allows users to publish short *tweets* (messages within a 140-character limit) about "what's happening". Real-life events are reported in *Twitter*. For example, the Iranian election protests in 2009 were extensively reported by *Twitter* users. Reporting those events could provide different perspectives to news items than traditional media, and also valuable user sentiment about certain companies/products.

This paper focuses on detecting those events to have a better understanding of what users are really discussing about in *Twitter*. *Event detection* has long been a research topic (Yang, Pierce, and Carbonell 1998). The underlying assumption is that some related words would show an increase in the usage when an event is happening. An event is therefore conventionally represented by a number of keywords showing *burst* in appearance count (Yang, Pierce, and Carbonell 1998; Kleinberg 2002). For example, "iran" would be used more often when users are discussing about the Iranian election protests. This paper also adapts such representation of *event*. Nevertheless, the characteristics of *Twitter* pose new challenges:

- The contents in *Twitter* are dynamically changing and increasing. According to *http://tweespeed.com*, there are more than 15,000 tweets per minute by average published in *Twitter*. Existing algorithms typically detect events by clustering together words with similar burst patterns. Furthermore, it is usually required to pre-set the number of events that would be detected, which is difficult to obtain in *Twitter* due to its real-time nature. A more scalable approach for event detection is therefore desired.

- Conventionally, *event detection* is conducted on formal document collections, e.g. academic papers (Kleinberg 2002) and news articles (Fung et al. 2005). It is assumed that all the documents in the collections are somehow related to a number of undiscovered events. However, this is not the case in *Twitter*, where tweets reporting big real-life events are usually overwhelmed by high flood of trivial ones. According to a study by Pear Analytics (Pear-Analytics 2009), about 40% of all the *tweets* are pointless "babbles" like "have to get something from the minimart downstairs". Such *tweets* are important to build a user's *social presence* (Kaplan and Haenlein 2010). Nevertheless, they are insignificant and should not require attention from the majority of the audience. It is therefore naive to assume that any word in *tweets* showing burst is related to certain big event. A good example is the popular hashtag "#musicmonday". It shows some bursts every Monday since it is commonly used to suggest music on Mondays. However, such bursts obviously do not correspond to an event that majority of the users would pay attention to. Event detection in *Twitter* is expected to differentiate the big events from the trivial ones, which existing algorithms largely fail.

To tackle these challenges, this paper proposes *EDCoW* (*E*vent *D*etection with *C*lustering of *W*avelet-based Signals), which is briefly described as follows. *EDCoW* builds signals for individual words which captures only the bursts in the words' appearance. The signals can be fast computed by *wavelet analysis* and requires less space for storage. It then filters away the trivial words by looking at their corresponding signal auto-correlations. *EDCoW* then measures the *cross correlation* between signals. Next, it detects the events by clustering signals together by *modularity*-based graph partitioning, which can be solved with a scalable

eigenvalue algorithm. To differentiate the big events from trivial ones, *EDCoW* also quantifies the events' *significance*, which depends on two factors, namely the number of words and the *cross correlation* among the words relating to the event.

In the rest of this paper, we first present a brief survey of relate work. Next, we give a concise description of wavelet analysis, before *EDCoW* is illustrated in detail. Experimental studies are also presented to show the performance of *EDCoW*. Finally, we conclude with directions for future work.

Related Work

Existing event detection algorithms can be broadly classified into two categories: *document-pivot* methods and *feature-pivot* methods. The former detects events by clustering documents based on the semantics distance between documents (Yang, Pierce, and Carbonell 1998), while the latter studies the distributions of words and discovers events by grouping words together (Kleinberg 2002). *EDCoW* could be viewed as a *feature-pivot* method. We therefore focus on representative *feature-pivot* methods here.

In (Kleinberg 2002), Kleinberg proposes to detect events using an infinite-state automaton, in which events are modeled as state transitions. Different from this work, Fung et al. model individual word's appearance as binomial distribution, and identify burst of each word with a threshold-based heuristic (Fung et al. 2005) .

All these algorithms essentially detect events by analyzing word-specific signals in the time domain. There are also attempts to analyze signals in the frequency domain. In (He, Chang, and Lim 2007), the authors apply *Discrete Fourier Transformation* (DFT), which converts the signals from the time domain into the frequency domain. A burst in the time domain corresponds to a spike in the frequency domain. However, DFT cannot locate the time periods when the bursts happen, which is important in event detection. (He, Chang, and Lim 2007) remedies this by estimating such periods with the Gaussian Mixture model.

Compared to DFT, *wavelet transformation* has more desirable features. *Wavelet* refers to a quickly varnishing oscillating function (Daubechies 1992; Kaiser 1994). Unlike the sine and cosine used in the DFT, which are localized in frequency but extend infinitely in time, wavelets are localized in both time and frequency domain. Therefore, wavelet transformation is able to provide precise measurements about when and to what extent bursts take place in the signal. This makes wavelet transformation a better choice for event detection, and is applied in this paper to build signals for individual words. It has also been applied to detect events from Flickr data in (Chen and Roy 2009).

There is recently an emerging interest in harvesting social intelligence from *Twitter*. For example, (Petrović, Osborne, and Lavrenko 2010) try to detect whether users discuss any new event that have never appeared before in *Twitter*. However, it does not differentiate whether the new event, if any, is trivial or not. In (Sakaki, Okazaki, and Matsuo 2010), the authors exploit tweets to detect critical events like earthquake. They formulate event detection as a classification problem. However, users are required to specify explicitly the events to be detected. And a new classifier needs to be trained to detect new event, which makes it difficult to be extended.

Wavelet Analysis

Wavelet analysis is applied in *EDCoW* to build signal for individual words. This section gives a brief introduction of related concepts.

Wavelet Transformation

The wavelet analysis provides precise measurements regarding when and how the frequency of the signal changes over time (Kaiser 1994). The wavelet is a quickly vanishing oscillating function. Unlike sine and cosine function of Fourier analysis, which are precisely localized in frequency but extend infinitely in time, wavelets are relatively localized in both time and frequency.

The core of wavelet analysis is *wavelet transformation*. *Wavelet transformation* converts signal from the time domain to the time-scale domain (scale can be considered as the inverse of frequency). It decomposes a signal into a combination of *wavelet coefficients* and a set of linearly independent basis functions. The set of basis functions, termed *wavelet family*, are generated by scaling and translating a chosen *mother wavelet* $\psi(t)$. Scaling corresponds to stretching or shrinking $\psi(t)$, while translation moving it to different temporal position without changing its shape. In other words, a wavelet family $\psi_{a,b}(t)$ are defined as (Daubechies 1992):

$$\psi_{a,b}(t) = |a|^{-1/2}\psi(\frac{t-b}{a}) \quad (1)$$

where $a, b \in \mathbb{R},\ a \neq 0$ are the scale and translation parameters respectively, and t is the time.

Wavelet transformation is classified into *continuous wavelet transformation* (CWT) and *discrete wavelet transformation* (DWT). Generally speaking, CWT provides a redundant representation of the signal under analysis. It is also time consuming to compute directly. In contrast, DWT provides a non-redundant, highly efficient wavelet representation of the signal. For (i) a special selection of the mother wavelet function $\psi(t)$ and (ii) a discrete set of parameters, $a_j = 2^{-j}$ and $b_{j,k} = 2^{-j}k$, with $j, k \in \mathbb{Z}$, the wavelet family in DWT is defined as $\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$, which constitutes an orthonormal basis of $L^2(\mathbb{R})$. The advantage of orthonormal basis is that any arbitrary function could be uniquely decomposed and the decomposition can be inverted.

DWT provides a non-redundant representation of the signal S and its values constitute the coefficients in a wavelet series, i.e. $< S, \psi_{j,k} >= Cj(k)$. $C_j(k)$ denotes the k-th coefficient in scale j. DWT produces only as many coefficients as there are sample points within the signal under analysis S, without loss of information. These wavelet coefficients provide full information in a simple way and a direct estimation of local energies at the different scales.

Assume the signal is given by the sampled values, i.e. $S = \{s_0(n)|n = 1, ..., M\}$, where the sampling rate is t_s and M is the total number of sample points in the signal. Suppose

that the sampling rate is $t_s = 1$. If the decomposition is carried out over all scales, i.e. $N_J = \log_2(M)$, the signal can be reconstructed by $S(t) = \sum_{j=1}^{N_J} \sum_k C_j(k)\psi_{j,k}(t) = \sum_{j=1}^{N_J} r_j(t)$, where the wavelet coefficients $C_j(k)$ can be interpreted as the local residual errors between successive signal approximations at scales j and $j+1$ respectively, and $r_j(t)$ is the detail signal at scale j, that contains information of the signal $S(t)$ corresponding with the frequencies $2^j\omega_s \leq |\omega| \leq 2^j\omega_s$.

Wavelet Energy, Entropy, and *H*-Measure

Since the wavelet family in DWT is an orthonormal basis for $L^2(\mathbb{R})$, the concept of energy derived from Fourier theory can also be applied (Adelson and Bergen 1985). The wavelet energy of signal S at each scale j ($j \leq N_J$) can be computed as:

$$E_j = \sum_k |C_j(k)|^2 \tag{2}$$

The wavelet energy at scale $N_J + 1$ can be derived as:

$$E_{N_J+1} = \sum_k |A_{N_J}(k)|^2 \tag{3}$$

The total wavelet energy carried by signal S is subsequently computed as follows:

$$E_{total} = \sum_{j=1}^{N_J+1} E_j \tag{4}$$

A normalized ρ-value measures the *relative wavelet energy* (RWE) at each individual scale j:

$$\rho_j = \frac{E_j}{E_{total}} \tag{5}$$

$\sum_{j=1}^{N_J+1} \rho_j = 1$. The distribution $\{\rho_j\}$represents the signal's wavelet energy distribution across different scales (Rosso et al. 2001).

Evaluating the Shannon Entropy (Shannon 1948) on distribution $\{\rho_j\}$ leads to the measurement of *Shannon wavelet entropy* (SWE) of signal S (Rosso et al. 2001):

$$SWE(S) = -\sum_j \rho_j \cdot \log \rho_j \tag{6}$$

SWE measures the signal energy distribution at different scales (i.e. frequency bands). *H*-Measure of signal S is defined as:

$$H(S) = SWE(S)/SWE_{max} \tag{7}$$

which is a normalized value of $SWE(S)$. SWE_{max} is obtained with a uniform distribution of signal energy across different scales, e.g. $\{\rho_j\} = \{\frac{1}{N_J+1}, \frac{1}{N_J+1}, \cdots \frac{1}{N_J+1}\}$.

EDCoW in Detail

This section details *EDCoW*'s three main components: (1) signal construction, (2) *cross correlation* computation, and (3) *modularity*-based graph partitioning.

Construction of Signals with Wavelet Analysis

The signal for each individual word (unigram) is built in two stages. Assuming T_c is the current time. In the first stage, the signal for a word w at T_c can be written as a sequence:

$$S_w = [s_w(1), s_w(2), \cdots, s_w(T_c)] \tag{8}$$

$s_w(t)$ at each sample point t is given by its *DF-IDF* score, which is defined as:

$$s_w(t) = \frac{N_w(t)}{N(t)} \times \log \frac{\sum_{i=1}^{T_c} N(i)}{\sum_{i=1}^{T_c} N_w(i)} \tag{9}$$

The first component of the right hand side (RHS) of Eq. (9) is *DF* (document frequency). $N_w(t)$ is the number of tweets which contain word w and appear after sample point $t-1$ but before t, and $N(t)$ is the number of all the tweets in the same period of time. *DF* is the counterpart of *TF* in *TF-IDF* (Term Frequency-Inverse Document Frequency), which is commonly used to measure words' importance in text retrieval (Salton and Buckley 1988). The difference is that *DF* only counts the number of tweets containing word w. This is necessary in the context of *Twitter*, since usually multiple appearances of the same word are associated with the same event in one single short tweet. The second component of RHS of Eq. (9) is equivalent to *IDF*. The difference is that, the collection size is fixed for the conventional *IDF*, whereas new tweets are generated very fast in *Twitter*. Therefore, the *IDF* component in Eq. (9) makes it possible to accommodate new words. $s_w(t)$ takes a high value if word w is used more often than others from $t-1$ to t while it is rarely used before T_c, and a low value otherwise.

In the second stage, the signal is built with the help of a sliding window, which covers a number of 1st-stage sample points. Denote the size of the sliding window as Δ. Each 2nd-stage sample point captures how much the change in $s_w(t)$ is in the sliding window, if there is any.

In this stage, the signal for word w at current time T'_c is again represented as a sequence:

$$S'_w = [s'_w(1), s'_w(2), \cdots, s'_w(T'_c)] \tag{10}$$

Note that t in the first stage and t' in the second stage are not necessarily in the same unit. For example, the interval between two consecutive t's in the first stage could be 10 minutes, while the interval in the second stage could be 60 minutes. In this case, $\Delta = 6$.

To compute the value of $s'_w(t')$ at each 2nd-stage sample point, *EDCoW* first moves the sliding window to cover 1st-stage sample points from $s_w((t'-2)*\Delta+1)$ to $s_w((t'-1)*\Delta)$. Denote the signal fragment in this window as $\mathcal{D}_{t'-1}$. *EDCoW* then derives the *H*-measure of the signal in $\mathcal{D}_{t'-1}$. Denote it as $H_{t'-1}$. Next, *EDCoW* shifts the sliding window to cover 1st-stage sample points from $s_w((t'-1)*\Delta+1)$ to $s_w(t'*\Delta)$. Denote the new fragment as $\mathcal{D}_{t'}$. Then, *EDCoW* concatenates segment $\mathcal{D}_{t'-1}$ and $\mathcal{D}_{t'}$ sequentially to form a larger segment $\mathcal{D}_{t^*}$, whose *H*-measure is also obtained. Denoted it as H_{t^*}. Subsequently, the value of $s'_w(t')$ is calculated as:

$$s'_w(t') = \begin{cases} \frac{H_{t^*} - H_{t'-1}}{H_{t'-1}} & \text{if } (H_{t^*} > H_{t'-1}); \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

If there is no change in $s_w(t)$ within $\mathcal{D}_{t'}$, there will be no significant difference between $s'_w(t')$ and $s'_w(t'-1)$. On the other hand, an increase/decrease in the usage of word w would cause $s_w(t)$ in $\mathcal{D}_{t'}$ to appear in more/less scales. This is translated into an increase/decrease of wavelet entropy in $\mathcal{D}_{t*}$ from that in $\mathcal{D}_{t'-1}$. And $s'_w(t')$ encodes how much the change is.

Figure 1 illustrates the two stages of signal construction in *EDCoW*. Figure 2 gives an example of the signals constructed based on tweets published by a number of Singapore-based *Twitter* users on June 16, 2010. On

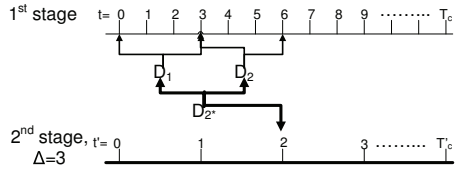
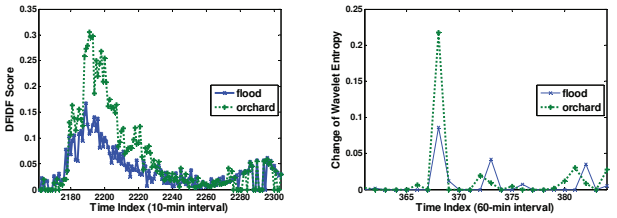


Figure 1: Two Stages of Signal Construction



(a) after first stage (b) after second stage

Figure 2: Example of Signals (2 stages)

that day, there was a heavy downpour in Singapore, which caused flash flood in the premium shopping belt Orchard road. At each sample point in Figure 2(a), $N_w(t)$ is the number of the tweets published in the past 10 minutes which contains the specific word, while $N(t)$ is the number of all the tweets published in the same period of time. Figure 2(b) is generated with $\Delta = 6$, i.e. one 2nd-stage sample point encodes the change of a word's appearance pattern in the past 60 minutes. Figure 2 shows that the bursts of the words are more salient in the corresponding 2nd-stage signals.

By capturing the change of a word's appearance pattern within a period of time in one 2nd-stage sample point, it reduces the space required to store the signal. In fact, event detection needs only the information whether a word exhibits any burst within certain period of time (i.e. Δ in the case of *EDCoW*). As we can see in Figure 2, 1st-stage signal contains redundant information about the complete appearance history of a specific word. Nevertheless, most existing algorithms store data equivalent to the 1st-stage signal.

After the signals are built, each word is then represented as its corresponding signal in the next two components[1].

Computation of Cross Correlation

EDCoW detects events by grouping a set of words with similar patterns of burst. To achieve this, the similarities between words need to be computed first.

[1]In the rest of this paper, "signal" and "word" are used interchangeably.

This component receives as input a segment of signals. Depending on the application scenario, the length of segment varies. For example, it could be 24 hours, if a summary of the events happened in one day is needed. It could also be as short as a few minutes, if a timelier understanding of what is happening is required. Denote this segment as $\mathcal{S}^{\mathcal{I}}$, and individual signal in this segment $S_i^{\mathcal{I}}$.

In signal processing, *cross correlation* is a common measure of similarity between two signals (Orfanidis 1996). Represent two signals as functions, $f(t)$ and $g(t)$, the *cross correlation* between the two is defined as:

$$(f \star g)(t) = \sum f*(\tau)g(t+\tau) \tag{12}$$

Here, $f*$ denotes the complex conjugate of f. Computation of *cross correlation* basically shifts one signal (i.e. g in Eq. (12)) and calculates the dot product between the two signals. In other words, it measures the similarity between the two signals as a function of a time-lag applied to one of them.

Cross correlation could also be applied on a signal itself. In this case, it is termed as *auto correlation*, which always shows a peak at a lag of zero, unless the signal is trivial zero signal. Given this, the *auto correlation* (with zero time lag) could be used to evaluate how trivial a word is. Denote signal $S_i^{\mathcal{I}}$'s *auto correlation* as $A_i^{\mathcal{I}}$.

Cross correlation computation is a pair-wise operation. Given the large number of words used in *Twitter*, it is expensive to measure *cross correlation* between all pairs of signals. Nevertheless, a large number of signals are in fact trivial. Figure 3 illustrates the distribution of $A_i^{\mathcal{I}}$ within $S_i^{\mathcal{I}}$ of 24-hour worth of signal. The distribution is highly skewed, i.e. the majority of the signals are trivial (with $A_i^{\mathcal{I}} \approx 0$). Given this, we discard the signals with $A_i^{\mathcal{I}} < \theta_1$. To set

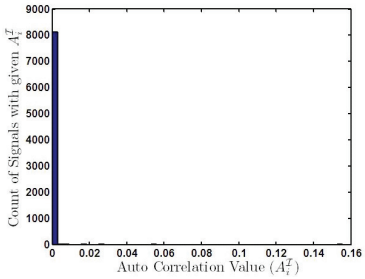


Figure 3: Skewed Distribution of Auto Correlation Values

θ_1, *EDCoW* first computes the *median absolute deviation* (MAD) of all $A_i^{\mathcal{I}}$ within $S_i^{\mathcal{I}}$:

$$MAD(\mathcal{S}^{\mathcal{I}}) = \text{median}(|A_i^{\mathcal{I}} - \text{median}(A_i^{\mathcal{I}})|) \tag{13}$$

MAD is a statistically robust measure of the variability of a sample of data in the presence of "outliers" (Walker 1931). In the case of *EDCoW*, we are interested in those "outliers" with outstandingly high $A_i^{\mathcal{I}}$ though. Therefore, we filter away those signals with $A_i^{\mathcal{I}} < \theta_1$, and θ_1 is set as follows:

$$\theta_1 = \text{median}(A_i^{\mathcal{I}}) + \gamma \times MAD(\mathcal{S}^{\mathcal{I}}) \tag{14}$$

Empirically, γ is not less than 10 due to the high skewness of $A_i^{\mathcal{I}}$ distribution.

Denote the number of the remaining signals as $\mathcal{K}$. *Cross correlation* is then computed in a pair-wise manner between

all the remaining $\mathcal{K}$ signals. Currently, the *cross correlation* between a pair of signals is calculated without applying time lag[2]. Denote the *cross correlation* between $S_i^{\mathcal{I}}$ and $S_j^{\mathcal{I}}$ as X_{ij}.

It is observed that the distribution of X_{ij} exhibits a similar skewness as the one shown in Figure 3. Given this, for each signal $S_i^{\mathcal{I}}$, *EDCoW* applies another threshold θ_2 on X_{ij}, which is defined as follows:

$$\theta_2 = \text{median}_{S_j^{\mathcal{I}} \in \mathcal{S}^{\mathcal{I}}}(X_{ij}) + \gamma \times MAD_{S_j^{\mathcal{I}} \in \mathcal{S}^{\mathcal{I}}}(X_{ij}) \quad (15)$$

Here, γ is the same as then one in Eq. (14). We then set $X_{ij} = 0$ if $X_{ij} \leq \theta_2$.

The remaining non-zero X_{ij}'s are then arranged in a square matrix to form the *correlation matrix* $\mathcal{M}$. Since we are only interested in the similarity between pairs of signals, the cells on the main diagonal of $\mathcal{M}$ are set to be 0. $\mathcal{M}$ is highly sparse after applying threshold θ_2. Figure 4 shows a portion of matrix $\mathcal{M}$ built from the data used in Figure 2. It shows the *cross correlation* between the top 20 words with the highest $A_i^{\mathcal{I}}$ on that day.

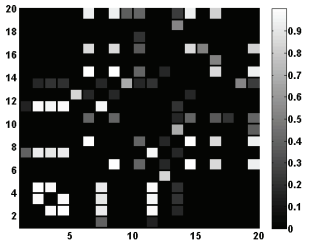


Figure 4: Illustration of Correlation Matrix $\mathcal{M}$. The lighter the color of the cell in the matrix, the higher the similarity between the two signals is, and vice versa.

The main computation task in this component is the pairwise *cross correlation* computation, which apparently has a time complexity of $\mathcal{O}(n^2)$, where n is the number of individual signals involved in the computation. n is generally very small after filtering with θ_1 (in Eq. (14)). For example, in the experimental studies, less than 5% of all the words remain after filtering with θ_1. The quadratic complexity is therefore still tractable.

Detection of Event by Modularity-based Graph Partitioning

Matrix $\mathcal{M}$ is a symmetric sparse matrix. From a graph theoretical point of view, it can be viewed as the adjacency matrix of a sparse undirected weighted graph $\mathcal{G} = (V, E, W)$. Here, the vertex set V contains all the $\mathcal{K}$ signals after filtering with *auto correlation*, while the edge set $E = V \times V$. There is an edge between two vertices v_i and v_j ($v_i,\ v_j \in V$) if $X_{ij} > \theta_2$, and the weight $w_{ij} = X_{ij}$.

With such a graph theoretical interpretation of $\mathcal{M}$, event detection can then be formulated as a graph partitioning problem, i.e. to cut the graph into subgraphs. Each subgraph corresponds to an event, which contains a set of words with high *cross correlation*. And the *cross correlation* between words in different subgraphs are expected to be low.

Newman proposes a metric called *modularity* to measure the quality of such partitioning (Newman 2004; 2006). The *modularity* of a graph is defined as the sum of weights of all the edges that fall within subgraphs (after partitioning) subtracted by the expected edge weight sum if edges were placed at random. A positive modularity indicates possible presence of partitioning. We can define node v_i's *degree* as $d_i = \sum_j w_{ji}$. The sum of all the edge weights in $\mathcal{G}$ is defined as $m = \sum_i d_i/2$. The *modularity* of the partitioning is defined as:

$$Q = \frac{1}{2m} \sum_{ij} (w_{ij} - \frac{d_i \cdot d_j}{2m}) \delta_{c_i, c_j} \quad (16)$$

where c_i and c_j are the index of the subgraph that node v_i and v_j belong to respectively, and δ_{c_i,c_j} is the Kronecker delta. $\delta_{c_i,c_j} = 1$ if $c_i = c_j$, or $\delta_{c_i,c_j} = 0$ otherwise.

The goal here is to partition $\mathcal{G}$ such that Q is maximized. Newman has proposed very intuitive and efficient spectral graph theory-based approach to solve this optimization problem (Newman 2006). It first constructs a *modularity matrix* (B) of the graph $\mathcal{G}$, whose elements are defined as:

$$B_{ij} = w_{ij} - \frac{d_i \cdot d_j}{2m} \quad (17)$$

Eigen-analysis is then conducted on the symmetric matrix B to find its largest eigenvalue and corresponding eigenvector ($\overrightarrow{\mathrm{v}}$). Finally, $\mathcal{G}$ is split into two subgraphs based on the signs of the elements in $\overrightarrow{\mathrm{v}}$. The spectral method is recursively applied to each of the two pieces to further divide them into smaller subgraphs.

Note that, with the modularity-based graph partitioning, *EDCoW* does not require extra parameter to pre-set the number of subgraphs (i.e. events) to be generated. It stops automatically when no more subgraph can be constructed (i.e. $Q < 0$). This is one of the advantages *EDCoW* has over other algorithms.

The main computation task in this component is finding the largest eigenvalue (and corresponding eigenvector) of the sparse symmetric modularity matrix B. This can be efficiently solved by *power iteration*, which is able to scale up with the increase of the number of words used in tweets (Ipsen and Wills 2006)

Quantification of Event Significance

Note that *EDCoW* requires each individual event to have at least two words, since the smallest subgraph after graph partitioning contains two nodes. This is rationale, since it is rare that a big event would only be described by one word if there are so many users discussing about it. Nevertheless, since each tweet is usually very short (less than 140 characters), it is not reasonable for an event to be associated with too many words either.

Given this, *EDCoW* defines a measurement to evaluate the events' *significance*. Denote the subgraph (after partitioning) corresponding to an event as $C = (V^c, E^c, W^c)$.

[2]As mentioned earlier, cross correlation measure the similarity between two signals as a time lag applied to one of them. By varying the time lag, it is possible to study the temporal relationship between two words, e.g. a word appears earlier than another in an event. We plan such study in future work.

V^c is the vertex set, $E^c = V^c \times V^c$, W^c contains the weights of the edges, which are given by a portion of *correlation matrix* $\mathcal{M}$. The event *significance* is then defined as:

$$\epsilon = (\sum w_{ij}^c) \times \frac{e^{1.5n}}{(2n)!},\ n = |V^c| \tag{18}$$

Eq. (18) contains two parts. The first part sums up all the *cross correlation* values between signals associated with an event. The second part discounts the *significance* if the event is associated with too many words. The higher ϵ is, the more significant the event is. Finally, *EDCoW* filters events with exceptionally low value of ϵ (i.e. $\epsilon \ll 0.1$).

Empirical Evaluation

To validate the correctness of *EDCoW*, we conduct an experimental study with a dataset collected from *Twitter*.

Dataset Used

The dataset used in the experiments is collected with the following procedure:

1. Obtain the top 1000 Singapore-based [3] *Twitter* users with the most followers from *http://twitaholic.com/*. Denote this set as U.
2. For each user in U, include her Singapore-based followers and friends within 2 hops. Denote this aggregated set as U^*.
3. For each user in U^*, collect the tweets published in June 2010.

Twitter REST API[4] is used to facilitate the data collection. There is a total of 19,256 unique users, i.e. $|U^*| = 19,256$. The total number of tweets collected is 4,331,937. The tweets collected are tokenized into words. Stop-words are filtered. We also filter (1) words with non-English characters, and (2) words with no more than three characters. Stemming is also applied. There are 638,457 unique words in total after filtering and stemming.

Experimental Settings

Before applying *EDCoW*, we further clean up the dataset. First of all, rare words are filtered, since they are less possible to be associated with an event. A threshold of five appearances every day by average is applied[5]. We further filter words with certain patterns being repeated more than two times, e.g. "booooo" ("o" being repeated 5 times) and "hahahaah" ("ha" being repeated 3 times). Such words are mainly used for emotional expression, and not useful in defining events. There are 8,140 unique words left after cleaning up.

To build signals for individual words, we set the interval between two consecutive 1st-stage sample points to be 10 minutes, and $\Delta = 6$. By doing so, the final signals constructed capture the hourly change of individual words' appearance patterns. *EDCoW* is then applied to detect events on every day in June 2010.

Correctness of *EDCoW*

In a typical information retrieval context, *recall* and *precision* are two widely used performance metrics. Given a collection of document, *recall* is defined as the fraction of the relevant documents retrieved to the total number of relevant documents should have been returned. In the case of *EDCoW*, "relevant" means there is a real-life event corresponding to the detected event. However, it is not feasible to enumerate all the real-life events happened in June 2010 in the dataset. It is therefore difficult to measure *EDCoW*'s *recall*. Given this, we concentrate on *precision* rather than *recall*, which measures the portion of the "relevant" events detected by *EDCoW* to all the events detected. Table 1 lists all the events (with $\epsilon > 0.1$) detected by *EDCoW*.

Since no ground truth is available about all the "relevant" events, we manually check the events detected by *EDCoW* one by one. There is no event (with $\epsilon > 0.1$) detected on June 1-3, 6, and 19-30. Out of the 21 events detected, we find three events which do not correspond to any real-life event, i.e. Event 6, 9, and 10 in Table 1. There is one event which is a mixture of more than one real-life event, i.e. Event 7. It is associated with two words, which correspond to two non-related real-life events. Event 13 is detected to associate with two words "#svk" and "#svn", which relate to two teams in the World Cup 2010. There was no clear real-life event related to the two teams on that day though. Therefore, the *precision* of *EDCoW* in this case is 76.2%.

EDCoW has one tunable parameter, i.e. γ in Eq. (14) and (15). The result so far is obtained with $\gamma = 40$. We also study *EDCoW*'s performance with different γ values, i.e. $\gamma = 10, 20, 30, 50$.

A smaller value of γ (i.e. $\gamma < 40$) fails to filter away signals with trivial *auto correlation*, many of which are included in the graph partitioning to form the events. In this case, most of the events detected by *EDCoW* are associated with a large number of words, and therefore small ϵ values. We also manually check the events detected by *EDCoW* with different γ values. None of the five events with $\epsilon > 0.1$ detected by *EDCoW* with $\gamma = 10$ corresponds to any real-life event. The *precision* in this case is 0. For $\gamma = 20$, only one out of seven events is "relevant", which corresponds to Event 3 in Table 1. This is translated to a *precision* of 14.3%. For $\gamma = 30$, only two out of 12 events are "relevant", which correspond to Event 2 and 3 in Table 1. The *precision* is therefore 16.7%.

A larger value of γ filters more signals away. In this case, some of the "relevant" events, if any, are already filtered before graph partitioning is applied to detect them. We again manually check the events detected. Although more events (with $\epsilon > 0.1$) are detected, only one new "relevant" event other than those listed in Table 1 is detected. It is associated with two words "ghana" and "#gha", and corresponds to a match between team Ghana and Serbia on June 13, 2010. There are another eight "relevant" events out of the total 40

[3] A user is considered Singapore-based if she specifies "Singapore" as her location in the profile.

[4] Twitter API: http://dev.twitter.com/doc#rest-api.

[5] To be consistent with Eq. (9), here we count the word appearance by the number of tweets using the word, even the word may appear in one single tweet more than once.

Day	Event	ϵ value	Event Description
1-3	No event detected		
4	1. democrat, naoto	0.417	Ruling Democratic Party of Japan elected Naoto Kan as chief.
	2. ss501, suju	0.414	Korean popular bands Super Junior's and SS501's performance on mubank.
	3. music, mubank	0.401	Related to Event 2, mubank is a popular KBS entertainment show.
	4. shindong, youngsaeng	0.365	Related to Event 2, Shindong and Youngsaeng are member of the two bands.
	5. junior, eunhyuk	0.124	Related to Event 2, Eunhyuk is a member of super junior.
5	6. robben, break	0.404	No clear corresponding real-life even
6	No event detected		
7	7. kobe, kristen	0.417	Two events: Kristen Stewart won some MTV awards, and Kobe Bryant in a NBA match.
	8. #iphone4, ios4, iphone	0.416	iPhone 4 released during WWDC 2010
8	9. reformat, hamilton	0.391	No clear corresponding real-life event
	10. avocado, commence, ongoing	0.124	No clear corresponding real-life event
9	11. #failwhale, twitter	0.360	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.
10	12. vuvuzela, soccer	0.387	People started to talk about world cup.
11	13. #svk, #svn	0.418	#svk and #svn represent Team Slovakia and Slovenia in World Cup 2010.
12	14. #kor, greec, #gre	0.102	A match between South Korea and Greece in World Cup 2010.
13	15. whale, twitter	0.417	Similar as Event 10.
14	16. lippi, italy	0.326	Italy football team coach Marcello Lippi made some comments after a match in World Cup 2010.
15	17. drogba, ivory	0.417	Football player Drogba from Ivory Coast is given special permission to play in World Cup 2010.
	18. #prk, #bra, north	0.114	A match between North Korea and Brazil in World Cup 2010.
16	19. orchard, flood	0.357	Flood in Orchard Road.
17	20. greec, #gre, nigeria	0.122	A match between Greece and Nigeria in World Cup 2010.
18	21. #srb, podolski	0.403	A match between Germany and Serbia in World Cup 2010. Podolski is a member of Team Germany in World Cup 2010.
19-30	No event detected		

Table 1: All the Events Detected by *EDCoW* in June 2010

detected events, which correspond to Event 1, 2, 3, 5 (with different words though), 7, 11, 13, and 20 in Table 1. The *precision* is 22.5%.

Comparison with Other Methods

In the experimental study, *EDCoW* is applied to detect the events on a daily basis. To some extent, this is equivalent to topic modeling, whose goal is to discover the "topics" that occur in a collection of documents. Given this, we aggregate all the tweets published on one day as one single document, and then apply topic modeling on the collection of documents (i.e. all the 30 documents for June 2010). We apply Latent Dirichlet Allocation (LDA), a widely used statistical topic modeling technique, on the document collection. We then compare the result generated from LDA with that by *EDCoW*.

In LDA, each document is a mixture of various topics, and the document-topic distribution is assumed to have a Dirichlet prior (with hype-parameter α). Each topic itself is a mixture of various words, and the topic-word distribution is again assumed to have a Dirichlet prior (with hype-parameter β) as well. LDA is conditioned on three parameters, i.e. Dirichlet hyper-parameters α, β, and topic number T[6]. In this study, they are set as $T = 50$, $\alpha = 50/T$ and $\beta = 0.1$. Due to the space constraint, the complete result of all the topics (each topic is represented as a list of top words) is omitted here. Instead, the top-4 topics identified on June 16, 2010 are listed in Table 2. The "probability" in this table is the probability that the corresponding topic appears in a document (i.e. all the tweets published on one particular day).

As it can be seen from Table 2, one of the obvious drawbacks of applying LDA in the context of event detection is that, the result generated by LDA is more difficult to interpret than the one listed in Table 1. Although "flood" and "orchard" are identified as the top words for the most related topic on June 16, 2010, they are mixed with other words as well. It is also not straightforward to see that Topic 8 may be related to "world cup". The other two top topics are even more difficult to interpret as their top-words are all trivial words. Moreover, after setting the number of topics (i.e. T), it would always return a distribution over T topics for each document no matter whether the document (i.e. tweets published on one particular day) has discussed about any real-life event or not. Further processing is required to improve the results generated by LDA in the context of event detection, e.g. applying threshold-based heuristics to filter non-eventful topics and words. In contrast, *EDCoW* has the ability to filter trivial words away before applying clustering technique to detect the events. More importantly, it requires no parameter to specify the number of events. It will automatically generate different number of events based on users' discussions in the tweets.

Conclusions and Future work

This paper focuses on detecting events by analyzing the contents published in *Twitter*. This paper proposes *EDCoW* (*E*vent *D*etection with *C*lustering of *W*avelet-based Signals). Experimental studies show that *EDCoW* achieves a fairly good performance. Nevertheless, *EDCoW* still has space for improvement.

First of all, currently *EDCoW* treats each word indepen-

[6]Due to space constraint, readers are referred to (Blei, Ng, and Jordan 2003) for the details of LDA.

Day	Topic ID	Probability	Top Words
16	13	0.229	flood, orchard, rain, spain, road, weather, singapor, love, cold
	48	0.095	time, don, feel, sleep, love, tomorrow, happi, home, hate
	11	0.091	time, love, don, feel, wait, watch, singapor, hope, life
	8	0.079	watch, world, cup, match, time, love, don, south, goal

Table 2: Topics Detected by LDA on June 16, 2010

dently. Such treatment may potentially group words associated with different real-life events together, as shown by the experimental study results. We plan to extend *EDCoW* by incorporating more factors, e.g. words need to be semantically close enough to be clustered to form an event. Second, we plan to study *EDCoW*'s performance with dataset of a larger scale. We also plan to investigate the possibility of compiling a ground truth automatically for the dataset, so that a more objective comparison with other algorithms could be conducted. Third, currently *EDCoW* does not exploit the relationship among users. It deserves a further study to see how the analysis of the relationship among users could contribute to event detection. Last but not least, the current design of *EDCoW* does not apply time lag when computing the *cross correlation* between a pair of words. We plan to introduce time lag and study the interaction between different words, e.g. whether one word appears earlier than another in one event. This could potentially contribute to study the temporal evolution of event.

Acknowledgements

We would like to thank Prof. Lim Ee-Peng from School of Information Systems, Singapore Management University for his valuable comments and discussion.

References

Adelson, E. H., and Bergen, J. R. 1985. Spatiotemporal energy models for the perception of motion. *Journal of Optical Society of America A* 2(2):284–299.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Chen, L., and Roy, A. 2009. Event detection from flickr data through wavelet-based spatial analysis. In *CIKM '09: Proceedings of the 18th ACM conference on Information and knowledge management*, 523–532. New York, NY, USA: ACM.

Daubechies, I. 1992. *Ten lectures on wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Fung, G. P. C.; Yu, J. X.; Yu, P. S.; and Lu, H. 2005. Parameter free bursty events detection in text streams. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, 181–192. VLDB Endowment.

He, Q.; Chang, K.; and Lim, E.-P. 2007. Analyzing feature trajectories for event detection. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 207–214. New York, NY, USA: ACM.

Ipsen, I. C., and Wills, R. S. 2006. Mathematical properties and analysis of google's pagerank. *Boletín de la Sociedad Espaňola de Matemática Aplicada* 34:191–196.

Kaiser, G. 1994. *A friendly guide to wavelets*. Cambridge, MA, USA: Birkhauser Boston Inc.

Kaplan, A. M., and Haenlein, M. 2010. The early bird catches the news: Nine things you should know about microblogging. *Business Horizons* To appear:–.

Kleinberg, J. 2002. Bursty and hierarchical structure in streams. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 91–101. New York, NY, USA: ACM.

Newman, M. E. J. 2004. Fast algorithm for detecting community structure in networks. *Physical Review. E* 69(6):066133.

Newman, M. E. J. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103(23):8577–8582.

Orfanidis, S. J. 1996. *Optimum Signal Processing*. McGraw-Hill.

PearAnalytics. 2009. Twitter study - august 2009. http://www.pearanalytics.com/wp-content/uploads/2009/08/Twitter-Study-August-2009.pdf.

Petrović, S.; Osborne, M.; and Lavrenko, V. 2010. Streaming first story detection with application to twitter. In *NAACL '10: Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Rosso, O. A.; Blanco, S.; Yordanova, J.; Kolev, V.; Figliola, A.; Schürmann, M.; and Başar, E. 2001. Wavelet entropy: a new tool for analysis of short duration brain electrical signals. *Journal of Neuroscience Methods* 105(1):65 – 75.

Sakaki, T.; Okazaki, M.; and Matsuo, Y. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW '10: Proceedings of the 19th international conference on World wide web*, 851–860. New York, NY, USA: ACM.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24(5):513 – 523.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27:623–656.

Walker, H. 1931. *Studies in the History of the Statistical Method*. Williams & Wilkins Co.

Yang, Y.; Pierce, T.; and Carbonell, J. 1998. A study of retrospective and on-line event detection. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 28–36. New York, NY, USA: ACM.