ET: Events from Tweets

Ruchi Parikh
International Institute of Information Technology, Hyderabad, India
ruchi.parikh@research.iiit.ac.in

Kamalakar Karlapalem
International Institute of Information Technology, Hyderabad, India
kamal@iiit.ac.in

ABSTRACT

Social media sites such as Twitter and Facebook have emerged as popular tools for people to express their opinions on various topics. The large amount of data provided by these media is extremely valuable for mining trending topics and events. In this paper, we build an efficient, scalable system to detect events from tweets (ET). Our approach detects events by exploring their textual and temporal components. ET does not require any target entity or domain knowledge to be specified; it automatically detects events from a set of tweets. The key components of ET are (1) an extraction scheme for event representative keywords, (2) an efficient storage mechanism to store their appearance patterns, and (3) a hierarchical clustering technique based on the common co-occurring features of keywords. The events are determined through the hierarchical clustering process. We evaluate our system on two data-sets; one is provided by VAST challenge 2011, and the other published by US based users in January 2013. Our results show that we are able to detect events of relevance efficiently.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval

Keywords

Tweets Processing, Event Detection, Text Anaytics, Hierarchical Clustering, Data Mining

1. INTRODUCTION

In recent years, netizens have started using social media services not only to interact with friends, but also to share their opinions and ideas about what they find interesting. Twitter, a micro-blogging service, provides such a facility to its users by allowing them to post 140 characters long 'tweets'. A recent study by an analyst group, Semiocast, states that Twitter has now passed the half-billion account mark as of July 1, 2012[1]. Very large numbers of tweets are posted daily on a variety of topics, ranging from casual chitchats to global news like elections and earthquakes.

1.1 Motivation

An important characteristic of Twitter is its real time nature. Its availability on web as well as mobile devices (cell phones) enables users to post tweets at any time and place. This ease of publishing messages on Twitter makes it a popular source of data to detect real-world events. There have been many events which were highlighted by Twitter users almost at the same time or before they were identified by the traditional news media. For example, Twitter revealed the 2010 cholera outbreak in Haiti two weeks before the health officials[2]. There are many applications that use Twitter to track earthquakes. The tragic earthquake that struck Japan in 2011 was identified within seconds because of the spikes in tweets from affected area, compared to the typical 2 to 20 minutes taken by scientific alerts[3]. Another prominent category of events detected by Twitter is sports. During the 2010 FIFA World Cup, Twitter observed huge traffic when a big goal was scored. Instead of 750 tweets per second on an average day, there were 2,940 tweets per second, after Japan scored against Cameroon[4]. Analysis of tweets related to such events gives a better understanding about user views and sentiments associated with the events.

Traditionally, events are detected from well-formatted, elaborately written documents like news articles and social streams like blogs, emails [18, 8, 3, 7]. These approaches can not be directly applied to tweets for three reasons. First, tweets are posted at a rapid rate, and thus produce a large amount of data, requiring scalable and efficient approaches. Secondly, because of the length restriction on tweets, the ideas are presented in brief and may not have enough information. Moreover, tweets are informally written and often contain grammatically incorrect text with misspellings and abbreviations. Thirdly, in most traditional approaches, it is assumed that all the documents are associated with some events, but that is not true with tweets. Tweets are inherently noisy and heterogeneous in nature. A study by Pear Analytics[5] states that half of the tweets are pointless and

ET with the latest results is demonstrated at `http://cdeproject.iiit.ac.in/et/`

[1]http://semiocast.com/publications/2012_07_30_Twitter_reaches_half_a_billion_accounts_140m_in_the_US

[2]http://mashable.com/2012/01/10/twitter-epidemic-choler-haiti/

[3]http://www.justmeans.com/Japan-Earthquake-on-Twitter-Social-Media-Trends-During-Disaster/46835.html

[4]http://www.huffingtonpost.com/2010/06/20/world-cup-2010-twitter-tr_n_617751.html

[5]http://www.pearanalytics.com/blog/2009/twitter-study-reveals-interesting-results-40-percent-pointless-babble/

do not convey any valuable information. These tweets are mainly related to personal updates of users, spam and self promotion. Processing such tweets not only increases the processing time, but also degrades the quality of results.

1.2 Overview and Features of our system

To deal with the above challenges, in this paper, we build a system, ET, to detect meaningful events from tweets efficiently. The important steps and features of our approach are as follows. Given a set of tweets, the time-span of this set is divided into fixed-length tunable time intervals, and tweets are segregated accordingly. Frequent keywords are found from these intervals, and keywords that show a sudden increase in their frequencies across consecutive time blocks are considered as event representative keywords. For each keyword, we maintain a list of the intervals in which it is frequent along with the frequency increases in those intervals. This list effectively captures the appearance pattern of a keyword, and is used to remove trivial bursty keywords. The event representative keywords that belong to the same event are grouped together using a hierarchical clustering technique. The similarity score between two keywords is defined using the common co-occuring textual features and similarity in their appearance patterns. The co-occuring textual features have to be computed only once for each keyword, and they are discovered only from the time blocks where the keyword is frequent. The appearance pattern of a keyword can be derived from the associated list of the keyword requiring no extra processing. These aspects make the clustering task very fast compared to most state-of-the-art techniques. At the end of the clustering process, each cluster represents one event. To make the events more interpretable and informative, we also use some frequent co-occuring textual features of the associated keywords. ET is able to detect events that cover a large variety of topics, ranging from small events like 'Car accidents' to big events like 'Golden Globes Awards' and 'Plane Crash'.

In the rest of the paper, we first provide an overview of related work (Section 2). In Section 3, we describe our system and its components in detail. Thereafter experiments and results are presented in Section 4, followed by conclusion.

2. RELATED WORK

The concept of event detection has its origin in the concept of topic detection. Since then many approaches have been proposed to mine events from news and broadcasting data. Most of the approaches [2, 1, 4, 17] used a document-pivot clustering technique to detect novel events from a stream of news stories. In contrast to this technique, feature-pivot clustering [3] first finds event related bursty features from the data/stream, and then applies clustering on these features to identify events. Fung et al. [3] represent an event as a minimal set of bursty features (unigrams). The bursty features are identified by their frequency distributions, which are modelled with binomial distributions. These features are grouped into bursty events by considering their probability of co-occurrence and frequency distributions. He et al. [5] use document frequency - inverse document frequency (df.idf) to build a signal for each feature in time domain. Discrete Fourier Transformation (DFT) is used to transform the signal in time domain to frequency domain. Features showing a spike in the frequency domain are extracted as bursty features.

While these approaches work well for well written articles, they do not work for tweets. Tweets are written informally with a lot of abbreviations and mistakes. Since tweets are very short, statistical concepts like $tf.idf$ can not be directly used on them [6].

Now, we consider recent work done in detecting events from tweets and other social media. Popescu et al. [13] propose a supervised classification method to decide whether, given a set of tweets involving an entity, the tweets focus on a single main event about that entity or not. An earthquake reporting system is built by Sakaki et al. [15] which uses the real-time nature of tweets to detect earthquake events. To detect tweets associated with the target event (earthquake), they use Support Vector Machine (SVM) classifier which uses the keywords and context of a tweet, as features. Twitris [12] is a semantic web application that uses tweets and their spatio-temporal features to gather information about real-world events. It also facilitates the browsing of related news and information using additional resources such as news feeds and Wikipedia. Lanagan et al. [9] use Twitter to automatically identify events of interest from a set of tweets generated at any moment of a live game of sports like football. TEDAS [11] detects Crime and Disaster related Events (CDE) from tweets using a CDE-focused crawler. They use spatial and temporal information of tweets to detect new events. While these papers find events pertaining to a specific entity like 'Julia Roberts','earth quake' or domain like 'sports', ET finds generic events which poses different set of challenges. TwiCal [14] extracts an open-domain calendar of significant events from Twitter and classifies the extracted events into important event categories. They employ an NLP-based approach to find named entities and event phrases from tweets. The extracted events are categorized into types based on latent variable models.

The two most related approaches to ET are EDCoW [16] and Twevent [10], that automatically detect generic events from tweets. EDCoW builds a signal for each individual word using wavelet theory to capture the burst in the word's appearance. It removes trivial words using their corresponding auto-correlations, and clusters the remaining signals based on the cross correlation between them using modularity based graph partitioning. While this approach yields good results, it lacks in scalability and computational efficiency. Building signals for each word and computing cross correlation for each pair of the signals takes a lot of computation time. Further, cross correlation only takes into account the similarity between the bursts of two words, without considering the similarity between the content associated with two words. This may cluster two events that are not related but happened in the same time span. Twevent [10] uses event segments constructed using the statistical information provided by Microsoft Web N-Gram service and Wikipedia, to present events. Thus the speed of segmentation relies on this service. The bursty segments are identified based on the tweet frequency of a segment. Out of these segments, noisy and trivial segments are removed manually. To find the similarity between the detected segments, Twevent divides the time window into sub-windows, computes frequencies of segments in each sub-window and, finds cosine similarity between sets of associated tweets for each sub-window. This amounts to a lot of additional computations, as computing frequencies of the segments in each sub-window requires an-

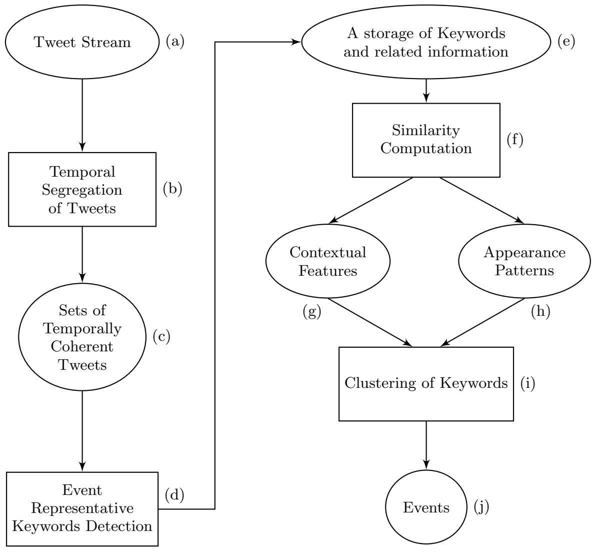

Tweet Stream
(a)
Temporal Segregation of Tweets
(b)
Sets of Temporally Coherent Tweets
(c)
Event Representative Keywords Detection
(d)
A storage of Keywords and related information
(e)
Similarity Computation
(f)
Contextual Features
(g)
Appearance Patterns
(h)
Clustering of Keywords
(i)
Events
(j)

Figure 1: ET Framework

other scan of the whole set of tweets. Similar segments are clustered using Jarvis-Patrick clustering algorithm. After clustering, they use anchor text in Wikipedia to re-rank the events such that more realistic events get more importance. This may give less importance to events like 'having fever' and 'bomb explosion', as they are not found in Wikipedia articles. Such events are actually of great importance to spread awareness.

ET uses bigrams as candidate keywords which reduces the computations caused by the large number of bursty unigrams and removes the bottleneck of using any N-Gram Service. Li et al. [10] found that over half of the segments are bigrams and there are rarely any segments of more than 3 grams. This observation supports our reasoning for using bigrams. Further, n-grams can be captured by ET while clustering the related bigrams based on their content similarity. We filter trivial bursty bigrams automatically without requiring any human intervention and extra computations. Unlike EDCoW, ET uses burst as well as content similarity to group related keywords. To compute this similarity, unlike Twevent, we divide the time window from the start, so frequencies need not be computed again. To compute content similarity, we first find co-occuring frequent features of each segment using only those time windows in which the segment is frequent. Once these features are obtained, the tweets need not be accessed again, and similarity is computed based on the common co-occuring features. This speeds up the entire process of clustering keywords.

3. ET SYSTEM

Figure 1 depicts our entire event detection framework. First component, Temporal Segmentation of Tweets (b), divides the tweet stream (a) into sets of temporally coherent tweets (c). Each set is processed to detect event representative bigrams (d). At the end of this process, we get a dictionary that stores these keywords and related information (e). This dictionary is used to compute content and appearance pattern similarity between these bigrams/keywords (f). These similarities are used as a distance metric in the hierarchical clustering technique (i). Each cluster in the resultant clustering represents an event (j). In the rest of the sections, we describe the main components of ET.

3.1 Event Representative Keywords

In this section, we present our model for extracting event representative keywords. We divide the total time span of tweets into fixed length time blocks, and notice the frequencies of keywords in each time block to identify an increase in their occurrences. The event representative keywords are the keywords which show a significant increase in their frequencies in a particular time interval. Consider the frequencies of a particular keyword in these intervals. If the difference of its frequencies in two consecutive time blocks is high, it can be inferred that some trend/event involving that keyword has started in the second time block. Given a set of tweets U, spanning over a time duration d, we divide d into a set of fixed length time blocks $\{d_1, \ldots, d_n\}$. The duration of each time block is fixed as b, and $n = d/b$. Accordingly, all the tweets in U are segregated into n non-overlapping subsets, $\{U_1, \ldots, U_n\}$, where U_i contains all the tweets that were written during time block d_i. The value of b can be set by users based on the kind of events they are interested in: short spanned or long spanned events. Typically for short spanned events, b is usually set to one day, but for long spanned events, it is set to one week.

As mentioned earlier, instead of using unigrams, we use bigrams as the candidate keywords. We start processing tweets from U_1 and go till U_n. Each tweet t in U_i is tokenized into words, and stop words are removed. Each pair of remaining consecutive words is taken as a bigram. For each such bigram k_j, we count the number of occurrences (absolute frequency) of k_j in U_i. After all the tweets in U_i are processed, we get the final values of absolute frequencies for all the bigrams in U_i. Now, the proportionate frequency of bigram k_j in interval U_i is calculated as follows:

$$PF(k_j) = \frac{AF(k_j)}{AF(k_1) + AF(k_2) + ... + AF(k_n)} \tag{1}$$

Here, $k_1, ...k_n$ are all the bigrams that are mentioned in interval U_j. $AF(k_j)$ and $PF(k_j)$ denote the absolute and proportionate frequency of k_j. If PF of a bigram k_j is greater than a threshold value γ, k_j is considered as a frequent bigram in U_i. We build an index in which, for each such bigram k_j, we maintain a list of pairs of the form $(U_i, h$. U_i represents the interval in which k_j is frequent and h shows the increase in frequency in this interval (i.e. difference between frequencies of k_j in U_i and U_{i-1}). To compute this difference, we store the frequency of k_j in the previous interval. If a bigram k is frequent in intervals U_1, U_3, and U_4 with increases h_1, h_3 and h_4 respectively, k is stored as follows:

$$k \rightarrow L[(U_1, h_1), (U_3, h_3), (U_4, h_4)] \tag{2}$$

Here, k is stored as a key in the index, while the list L of pairs is stored as the value of k. Note that a pair is added to the list only if k_j is frequent in the current interval. This way we reduce the storage requirements while maintaining the appearance pattern of a keyword. The second component of the pair, h, shows the increase of the frequency in the current interval. If it is high, it means that there is a sudden increase in the frequency and may indicate an event. If h is very low for all the frequent intervals of a keyword, it means that the keyword does not show a significant increase in any of them. These keywords do not represent an event, and are considered as trivial keywords. Processing such trivial keywords not only adds unnecessary computations, but also diminishes the accuracy of the system. We are able to filter such keywords without requiring any extra precessing. A candidate keyword should have at least one frequent interval showing a high increase. Such keywords are deemed event representative keywords. The index/storage mechanism described above is also utilized in the next section for clustering similar event representative keywords.

3.2 Clustering of Keywords

Once we get a set of event representative keywords, the next step is to cluster the keywords that are related to the same event. It can be said that two keywords belong to the same topic, if they are associated with similar content. But to say that they belong to the same event, it is important to see that they follow similar appearance patterns too. Our similarity metric considers both of these aspects.

3.2.1 Content Similarity

We compute the content similarity between two keywords by finding similarity between the sets of their associated tweets. In most approaches, the whole dataset is considered to find these sets of tweets. But, in our approach, we consider tweets from only the time blocks in which the keywords are frequent. It is intuitive that most of the tweets, which are related to the event, belong to the frequent time blocks of the keyword. Thus, considering only frequent time blocks reduces a lot of computations, while maintaining the quality of the results.

To gather such set of tweets for a keyword k_1, we utilize the index built in the previous section to find the list of frequent blocks of k_1, and gather tweets from these blocks. From this set of tweets, we find a set of bigrams that frequently co-occur with k_1. To find such frequently co-occuring bigrams (FCB) of k_1, we compute proportionate frequencies of all the bigrams that co-occur with k_1. The bigrams that have frequencies higher than a threshold value are collected and added to the set $FCB(k_1)$. The same procedure is repeated to find FCBs for each keyword. Thereafter, the content similarity between a pair of keywords is found by just using the FCBs, without going through the associated tweets for each pair again and again. The main bottleneck in computing similarity is the access time to go through the tweets. Since we access the tweets only once, the procedure becomes very efficient. The content similarity between two keywords k_1 and k_2 is defined using the Jaccard similarity coefficient.

$$C\_Sim(k_1, k_2) = \frac{FCB(k_1) \cap FCB(k_2)}{FCB(k_1) \cup FCB(k_2)} \tag{3}$$

$C\_Sim(k_1, k_2)$ focuses on the common frequently co-occuring bigrams between k_1 and k_2. Higher $C\_Sim(k_1, k_2)$ means that k_1 and k_2 are content similar. FCBs effectively capture the main contextual features of a keyword. Since they are found frequent in the time blocks wherein the keywords are frequent, there are high chances that they are also related to the event. This way, our similarity measure ensures that keywords that are similar are not just textually similar, but are similar in the context of the event.

3.2.2 Similarity of Appearance Patterns

For two keywords to be related to the same event, it is important that they follow similar appearance patterns. Our intuition here is that, if two keywords are frequent in the same set of time blocks, they follow closely similar frequency patterns, and thus may relate to the same event. While making the keyword index, for each keyword, we maintained a list of pairs of the form - [frequent interval, increase]. Thus, no further processing is required to find the common frequent time intervals (FI) of two keywords.

The appearance similarity between keywords k_1 and k_2 is defined as follows:

$$A\_Sim(k_1, k_2) = \frac{FI(k_1) \cap FI(k_2)}{FI(k_1) \cup FI(k_2)} \tag{4}$$

Here also, we use Jaccard coefficient to find common frequent time intervals. For each keyword, we find the frequent interval with the highest increase. We call this interval as 'start interval', as it is the interval when event has started. If two keywords have the same start time interval, they are given high similarity score.

3.2.3 Overall Similarity

Once we have both appearance and content similarities, the overall similarity between two keywords can be found as follows:

$$Sim(k_1, k_2) = \alpha \cdot C\_Sim(k_1, k_2) + \beta \cdot A\_Sim(k_1, k_2) \tag{5}$$

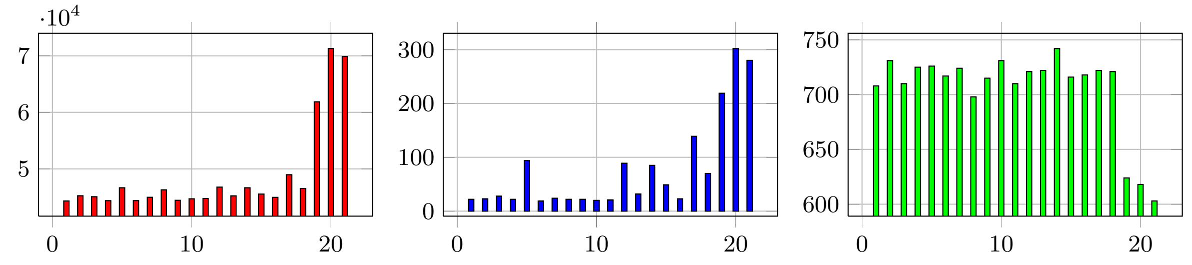
Here, the values of α and β can be tuned to vary the impact of content similarity and appearance pattern similarity on the overall score. Usually, the value of α is kept higher than the value of β to give more importance to the content similarity. We show some experiments with different values of α and β in the next section.

3.2.4 Hierarchical Clustering

We cluster the keywords using an agglomerative hierarchical clustering technique. The algorithm does not require the number of clusters to be specified before hand; just a similarity threshold has to be specified. Once we compute the similarity between each pair of keywords, we use it to build a similarity matrix. The algorithm starts with merging the keywords with the highest similarity score. After merging the keywords k_1 and k_2, we compute the similarity score between this cluster (new_clust) and any other keyword k_i as follows:

$$Sim(new\_clust, k_i) = \max\{Sim(k_1, k_i), Sim(k_2, k_i)\} \tag{6}$$

Accordingly, the similarity matrix is updated. The rows for keywords k_1 and k_2 are removed from the matrix, and a new row for the merged cluster is added to the matrix. This process is repeated till there is a keyword/cluster pair with similarity score higher than the threshold. When the process stops, we get the resultant clustering, where each cluster represents a set of keywords associated with one event. The resultant number of clusters and the size of the clusters vary according to the value of the similarity threshold. Flat clustering algorithms like *k-means*, do not work effectively in this case, as it is difficult to predict the number of clusters or the value of k beforehand. Further, the results of such algorithms are different for different initial patterns and values of k. These algorithms are prone to converge at a poor



(a) Tweets (b) Frequent Bigrams (c) Frequent Unigrams

Figure 2: Day-wise Frequency of Tweets, Bigrams and Unigrams

local optima if the initial seeds are not selected properly[6]. Hierarchical clustering fits here because of its deterministic and predictive behaviour. The resultant clustering only depends on the similarity threshold, which can be varied by the user to analyse the data. Once we have the clusters/events, we rank them in the descending order of the size the cluster. The cluster with the maximum number of keywords is ranked at the first position.

4. EXPERIMENTS AND RESULTS

In this section, we give details about the datasets used, experimental settings and a comprehensive analysis on the obtained results. We evaluate the impact of using unigrams and bigrams as candidate keywords on the results, and show that bigrams are more effective and efficient. Further, we show the importance of content similarity and appearance pattern similarity on the clustering. We show that the precision of the algorithm is better than that of the existing systems and the running time of this algorithm is less than that of most state-of-the-art systems.

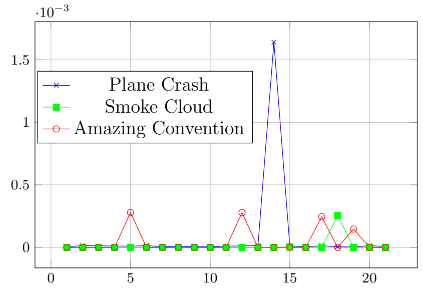
4.1 Dataset Description

We use two datasets for the evaluation of ET. The first dataset is provided by the VAST 2011 challenge. The dataset contains a total of 1023077 microblogs posted by users in May 2011. It contains microblog messages collected from various devices with GPS capabilities. The second dataset contains 33579 tweets collected during a one week period from January 13 to January 19, 2013. These tweets are associated with the broadcasting domain, and contain information about various TV shows, movies, games, popular videos and songs.

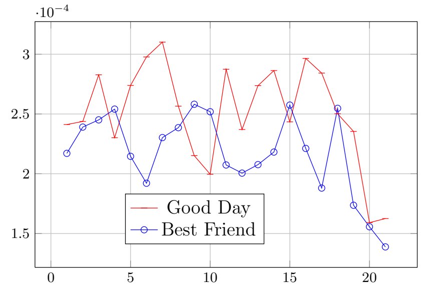
4.2 Experiment settings

The performance of our system and the resultant events depend on some parameters: first is the time block value. The messages in VAST dataset span over 20 days, so we set the value of each time block as one day. For broadcasting dataset, we set it to 6 hours. Thus, each day is divided into 4 time blocks. The second parameter is the frequency threshold γ to get the frequent bigrams. The value of this parameter depends on the frequency distributions of keywords. From the distributions, we find the most prominent frequency range for the keywords, and set the value of this threshold to a slightly higher value than this range. The

[6]http://nlp.stanford.edu/IR-book/html/htmledition/references-and-further-reading-17.html#sec:hclstfurther



(a) Event Representative Keywords



(b) Trivial Keywords

Figure 3: Frequency Distribution of Trivial and Event related Keywords

third parameter includes the values of α and β which indicate the weightages of content and appearance similarities respectively. These values have considerable impact on the resultant clustering, and have to be set carefully. Users can analyze the results by changing these values and see the impact. This analysis helps in understanding the events, their content and their appearance patterns. We here show the results for three different combinations of α and β.

4.3 Evaluation of Detected Keywords

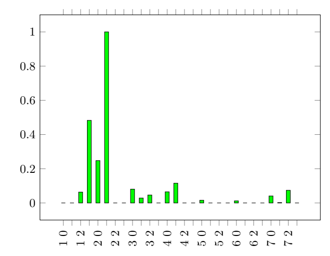
VAST Dataset: Fig 2(a) shows the number of tweets published on each day. It can be clearly seen from the plot that the tweets posted on the last 3 days are considerably more than those on other days. In Fig 2(b) and Fig 2(c), we have plotted the number of frequent bigrams and the number of frequent unigrams, detected on each day. These figures clearly depict the advantages of using bigrams as event representative keywords and not unigrams. The daily count is much higher for unigrams than it is for bigrams. This results in a huge amount of computation, and because of many false positives, it also degrades the quality of the resultant events. Further, it can be seen from Fig 2(a) and Fig 2(b) that both bigrams and tweets follow a similar pattern. The days when more tweets are published, the numbers of frequent bigrams are also high. Unigrams on the other hand, follow a completely random pattern. The number of detected unigrams are almost in the same range every day.

ET detected 902 unique frequent bigrams at the end of processing tweets from all the days. Keywords which were found frequent in most of the intervals, showing no considerable increase in any of them were removed by the system. ET removed 10 such keywords automatically. Fig 3(b) shows the frequency distributions of such removed keywords. The keywords 'Best Friend' and 'Good Day' are frequent in almost all the intervals and the frequencies only show a slight deviation from the frequency threshold for all the days. They show no significant increase on any day. Fig 3(a) shows the frequency distributions of some retained event representative keywords. These keywords show high frequencies only on particular days, indicating the existence of events on these days. Keywords 'Plane Crash' and 'Smoke Cloud' show a sudden increase on the days 14 and 18 respectively. Keyword 'Amazing Convention' has high frequencies on 3 days - 5, 12, 17. After the removal of trivial keywords, we cluster the remaining 892 event representative keywords.

Broadcasting Dataset: We divide this dataset into 6 hour windows. For each day, we get four intervals(0 to 3), totalling to 28 intervals. Fig 4(a) shows the tweets distribution across all these intervals. The X axis labels show [day, interval] pairs. Pair $(1, 0)$ indicates 1^{st} day and 0^{th} interval, i.e. the first 6 hours of day 1. The figure shows that most of the tweets are posted on last 6 hours of day 1 (Jan 13/Sunday) and first 12 hours of day 2 (Jan 14/Monday). This is because a lot of events happened on Sunday like Golden Globes Awards, Super bowl games, etc., and people are writing about it during the next 12 hours. Fig 4(b) shows the frequent bigrams detected in each interval. They follow almost the same distribution as tweets except for one change. The Numbers of bigrams detected from intervals $(1, 3)$ and $(2, 1)$ are almost same, but the number of tweets posted in $(1, 3)$ is much less than that of $(2, 1)$. Though number of tweets in interval $(2, 1)$ is too high, these tweets cover less distinct events compared to what the tweets in $(1, 3)$ cover. We detected a total of 116 frequent bigrams from all the intervals. Out of these bigrams, 15 bigrams are found trivial and are removed by ET. Some examples of such bigrams are 'watch tv', 'watch movies', 'laying bed', 'fall asleep', etc. The remaining 101 event representative bigrams are clustered to find events.

4.4 Evaluation of Clustering

Table 1 and Table 2 show the final clusters of related event representative keywords for the broadcasting dataset and the VAST dataset respectively. To evaluate how interpretable the detected keywords are, we gave these clusters of keywords to humans to interpret the events. Such interpreted events are also provided in the table. We also provide the time intervals on which these events happened. From broadcasting dataset, we have detected events like 'Golden Globes Awards', 'Super Bowl', 'Sandy Hook Controversy', 'Bad Girls Club Atlanta Season 10', etc. An event like


(a) Tweets

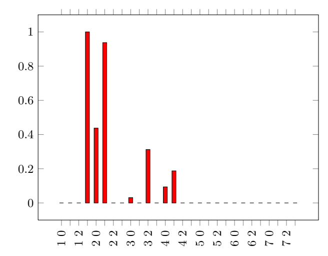

(b) Event Representative Bigrams

Figure 4: Normalized frequency of Tweets and Bigrams on each (Day, Interval) pair for Broadcasting Data

'Downton Abbey wins at Golden Globes' is also captured by ET. From VAST dataset, events like 'Plane crash', 'Disease Outbreak', 'Bomb threats', 'Conventions' and 'Car accidents', are detected.

Table 3: Frequently Co-occuring Bigrams

Plane Crash	crash landed, airport better, better protocols, plane flames, airport fire
Bomb Threats	bombs city, threats rampant, bomb threats, hope people, people alright
Sandy Hook	hook conspiracy, hook shooting, conspiracy video

Resultant clustering of the related keywords depends on the content similarity and appearance pattern similarity values between keywords. To get the content similarity, we have to find frequent co-occuring features/bigrams for each keyword. We consider only the top 25 most frequent features. Keywords having many common frequent features are content-related to the same event. In Table 3, we show frequent bigrams for the keywords 'Plane Crash' and 'Bomb Threats'.

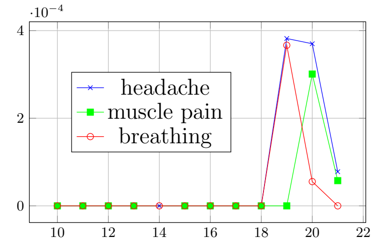


Figure 5: Appearance pattern for event 'Disease Outbreak'

Most of these bigrams are actually the main event keywords. This happens because most of the event keywords co-occur extensively which is captured by these bigrams. There are rarely any frequent bigrams which are not related to the event/event keyword. In case of the event 'Bomb Threats', we found bigrams like 'hope people', 'people alright'. These keywords without any context seem trivial, but in the context of this event, they depict the sentiments/emotions of people, and are important to analyze an event. Moreover if an event has only one or two event representative keywords, frequent co-occuring bigrams of these keywords are added to the keywords list to better understand the event. For example, the event 'Sandy Hook video' has only one event representative keyword - 'Sandy Hook'. It is not clear from this one keyword what the event is about. Hence, frequent co-occuring bigrams of 'Sandy Hook' are added to the keyword list of this event.

Table 1: Events Detected from Broadcasting Dataset

Event Representative Bigrams	Event	Time Interval(Day)
dense fog, advisory issued, rip current, statement issued, issued january, nws melbourne	Dense Fog Advisory and Rip Current Statement issued by NWS Melbourne	January 14, Interval-1
super bowl, tom brady, football game, playoff games, seahawks game, falcons seahawks, falcons game, dirty birds, nfl playoffs	Super bowl football nfl playoffs, Falcons (Dirty Birds) vs Seahawks game, People excited to watch Tom Brady	January 13, Interval-3
golden globes, globes tonight, downton abbey, red carpet,	Golden Globes Awards, Red Carpet, Maggi Smith of Downton Abbey (TV Series) wins at Golden Globes Awards.	January 14, Interval-0
season bad, bad girls, girls club, club atlanta	Bad Girls Club Atlanta Season 10 premiered on Jan 15	January 16, Interval-0, 1
patriots game, pats game, texans game	Patriots game Vs Texans game	January 13, Interval-3
sandy hook, hook conspiracy, hook shooting, conspiracy video	People watching Sandy Hook shooting conspiracy video.	January 14, Interval-1; January 16, Interval-1
music video, video macmerk, macmerk feat, feat yungking	Latest music video by Mac Merk Feat. Yung King	January 15, Interval-2
dark thirty, gangster squad, haunted house	Three movies released on same date: Zero Dark Thirty, Gangster Squad, and A Haunted House	January 13, Interval-3; January 14, Interval-0

Fig 5 shows the appearance pattern of some of the keywords for the event 'Disease Outbreak'. The keyword 'headache' is frequent on 3 days: 19, 20, 21, keyword 'muscle pain' is frequent on days 20 and 21, and 'breathing' is frequent on days 19 and 20. These keywords follow partial similar patterns but are grouped as one event. On the other hand, in Fig 6, for the event 'Super Bowl', the keywords 'football game', 'patriots game' and 'super bowl' follow very similar patterns. They are frequent in the same interval.

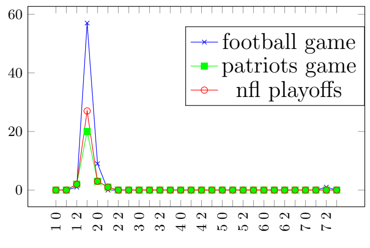


Figure 6: Appearance pattern for event 'Super Bowl'

Importance of content and appearance similarities: The results shown in Table 1 and Table 2 are obtained by setting the values of α and β 0.6 and 0.4 respectively. It means that slightly more weightage is given to content similarity than appearance similarity. Now, we change the value of α to zero and that of β to one. This means that we consider only appearance similarity. The resultant clustering for the VAST dataset shows the problem of duplicate events: There were four different clusters detected for the event 'Disease Outbreak'. Most of the keywords in these clusters describe the same symptoms. Keywords 'bad diarrhea' and 'diarrhea annoying' are put in different clusters as one is more frequent on day 20, and the other on day 21. On the other hand, for the broadcasting dataset, three different events 'Super bowl', 'Sandy Hook' and 'Golden Globes Awards' are clubbed into one cluster, as they are frequent in the same time interval - day 1, interval 3. Content Similarity helps in alleviating both of these problems.

To understand the impact of appearance pattern similarity, we now take α as one and β as zero. The keywords are clustered only on the basis of their content similarity. Different events related to the same topic get clubbed together. For the VAST dataset, events 'Convention', 'Sham Wow Convention', 'Antique Convention' and 'Comic Convention' in Table 2 are clubbed into one cluster, and presented as a single event. These events have a lot of common frequent content features about the topic 'convention', but they follow completely different appearance patterns. In some cases, different text/words are used to describe the same event, and this may result in multiple clusters for one event. This happened for the event 'Super Bowl'. We got three different clusters for this event. One cluster contains keywords about 'Falcons game', one about 'NFL playoffs' and other about 'Super Bowl and football'. However, when we also consider the appearance patterns of these keywords, they get clubbed into one cluster, as shown in Table 1. Therefore, only a combination of content and appearance similarities yields accurate results.

Evaluation Metric: We use the same definitions of precision and recall as defined by Weng et al. [16]. Since the dataset used by EDCoW and Twevent is not available, we have evaluated ET on different datasets. Recall, defined as the number of relevant events, can not be used as an evaluation matrix, as it varies for different datasets. We use precision as an evaluation matrix, as it captures the accuracy of the system. Since there is no ground truth available, we evaluate our results manually based on the knowledge of real world events obtained from general search. For the VAST dataset, ET detected a total of 23 events, out of which 2 events were trivial and insignificant. Thus, the precision of the algorithm is 91% and the recall is 21. For the broadcasting dataset, ET obtained a total of 15 events, out of which only 1 event was not related to any realistic event, giving the precision of 93% and recall of 14. Our experiments are conducted on a machine with 2.13GHz Intel Core i3 CPU and 4GB of RAM. ET takes about 157 seconds to detect events from VAST dataset, processing about 1023K tweets. This is quite efficient considering the specifications of our machine.

5. CONCLUSIONS

We present a scalable and efficient system, called ET, to detect real world events from a set of microblogs/tweets. The key feature of this system is the efficient use of content similarity and appearance similarity among keywords, to cluster the related keywords. We demonstrate the effectiveness of this combination in our experiments. ET does not

Table 2: Events Detected from VAST Dataset

Event Representative Bigrams	**Event**	**Time Interval(Day)**
headache causing, cold cough, fever sick, fatigue beg, muscles dreadful, pneumonia hurts, chills hurts, diarrhea annoying, upset stomach, extremely sick	Disease Outbreak, People suffering from severe disease.	19, 20, 21
plane submerged, submerged ground, airport fire, destruction airport, dammage airport, death people, people inside	Plane crash on airport, death of people inside.	14
threats reported, buildings blow, bomb crews, crews gearing, bomb threat, police searching, searching bombs, emergency crews,	bomb threats reported, police and emergency crews are searching for the bombs.	17
smog town, town explosion, ground shake, big boom, massive smoke, scary stuff, heard blast, huge smoke, smoke cloud,	Smog Town explosion, huge smoke cloud.	18
cool convention, total bargain, missing stuff, large attendance, pretty crowded, peeps convetion, heading convetion, things expensive, great deals	People excited about going to convention	5, 12, 17, 19
building terrible, capital building, scene panic, destruction fire, terrible scene, buildings burning, burning ashes, chaotic fire, building unsafe, fire crew, crew arrives,	Huge fire in the buildings near the capital building.	15
fascinated antique, antique amazing, antique large, antique awesome, antique convention	Antique Convention	5
fascinated sham, sham awesome, sham place, sham convention,	Sham Wow Convention	12
better comic, comic everyinthing, comic pretty, comic convention	Comic Convention	17
bad traffic, traffic move, car accident, terrible destruction	Bad Traffic and Car accidents	3, 7, 13, 18
terrible car, destruction car, traffic car	Terrible car accident	13

need any human expertise or knowledge from other sources like Wikipedia, and still provides very accurate results. ET is evaluated on two different datasets from two different domains and it yields great results for both of them in terms of the precision. In future, we will use a semantic knowledge base like Yago to further improve the quality of the results.

6. 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 and F. Chen. A System for new event detection. In *SIGIR*, pages 330–337, 2003.

[3] 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.

[4] V. Guralnik and J. Srivastava. Event detection from time series data. In *KDD*, pages 33–42, 1999.

[5] Q. He, K. Chang, and E.-P. Lim. Analyzing Feature Trajectories for Event Detection. In *SIGIR*, pages 207–214, 2007.

[6] D. Inouye and J. K. Kalita. Comparing Twitter Summarization Algorithms for Multiple Post Summaries. In *SocialCom/PASSAT*, pages 298–306, 2011.

[7] J. Kleinberg. Bursty and Hierarchical Structure in Streams. In *KDD*, pages 91–101, 2002.

[8] G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *SIGIR*, pages 297–304, 2004.

[9] J. Lanagan and A. F. Smeaton. Using Twitter to Detect and Tag Important Events in Live Sports. *AAAI*, 29(2):542–545, 2011.

[10] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *CIKM*, pages 155–164, 2012.

[11] R. Li, K. H. Lei, R. Khadiwala, and K. Chang. TEDAS: A Twitter-based Event Detection and Analysis System. In *ICDE*, pages 1273–1276, 2012.

[12] M. Nagarajan, K. Gomadam, A. P. Sheth, A. Ranabahu, R. Mutharaju, and A. Jadhav. Spatio-Temporal-Thematic Analysis of Citizen Sensor Data: Challenges and Experiences. In *WISE*, pages 539–553, 2009.

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

[14] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open domain event extraction from twitter. In *KDD*, pages 1104–1112, 2012.

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

[16] J. Weng and B. Lee. Event Detection in Twitter. In *ICWSM*, 2011.

[17] 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, 1999.

[18] Q. Zhao and P. Mitra. Event Detection and Visualization for Social Text Streams. In *ICWSM*, pages 26–28, 2007.