Journal of Computational Science 6 (2015) 47–57

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

# Cluster-discovery of Twitter messages for event detection and trending

Shakira Banu Kaleel*, Abdolreza Abhari

*Department of Computer Science, Ryerson University, Toronto, Canada*

ARTICLE INFO

*Article history:*
Received 3 December 2013
Received in revised form 22 August 2014
Accepted 18 November 2014
Available online 29 November 2014

*Keywords:*
Web social networking
Data mining
Event detection
Event trending
Locality sensitive hashing
Cluster-discovery
Term frequency–inverse document frequency

ABSTRACT

Social media data carries abundant hidden occurrences of real-time events. In this paper, a novel methodology is proposed for detecting and trending events from tweet clusters that are discovered by using locality sensitive hashing (LSH) technique. Key challenges include: (1) construction of dictionary using incremental term frequency–inverse document frequency (TF–IDF) in high-dimensional data to create tweet feature vector, (2) leveraging LSH to find truly interesting events, (3) trending the behavior of event based on time, geo-locations and cluster size, and (4) speed-up the cluster-discovery process while retaining the cluster quality. Experiments are conducted for a specific event and the clusters discovered using LSH and K-means are compared with group average agglomerative clustering technique.

## 1. Introduction

Social media provides an abundance of data on public opinions which can be used to extract the occurrences of real-time events in the world. The value of content generated on social media attracts people toward them [1]. The increase in popularity of social media websites such as Twitter,[1] Facebook,[2] LinkedIn,[3] YouTube[4] and Pinterest[5] accumulate the large amount of user-contributed data on the web. The first systematic work concerning the topic and event detection in the newswire document corpus was conducted during the topic detection and tracking (TDT) research. TDT research focuses on finding techniques to identify event documents from a group of news documents. There are five different tasks of a TDT system [2]:

- *Story segmentation*: The process of dividing news into individual stories.
- *First story detection (FSD) or new event detection (NED)*: The stories are monitored to detect the events which have not been seen before, also known as new event detection.
- *Cluster detection*: Similar stories are grouped under a specific topic.
- *Tracking*: News streams are monitored for additional stories which are related to previously existing stories.
- *Story link detection*: The link between stories discussing the same topic of interest is identified.

These tasks combined together to detect new events occurring around the globe. Each task of TDT system has caused a separate research problem in the field of information retrieval. According to Fiscus and Doddington [3], the definition of topic, story and event are explained as follows:

> Topic: "A seminal event or activity, along with all directly related events and activities."
>
> Story: "A topically cohesive segment of news that includes two or more declarative independent clauses about a single event."
>
> Event: "Something that happens at specific time and place along with all necessary conditions and unavoidable consequences."

* Corresponding author. Tel.: +1 416 979 5000.
*E-mail address:* kshakira@ryerson.ca (S.B. Kaleel).

[1] https://twitter.com/
[2] https://www.facebook.com
[3] https://www.linkedin.com
[4] https://www.youtube.com
[5] http://www.pinterest.com

http://dx.doi.org/10.1016/j.jocs.2014.11.004
1877-7503/© 2014 Elsevier B.V. All rights reserved.

For example, "Federal Election" is a topic whereas "Canada Federal Election 2015" is an event. The various discussions about the federal election are stories related to that topic.

Social media data captures the contents associated with various types of events including both planned (e.g. presidential elections and concerts) and unplanned (e.g. natural disaster and epidemics) events. Twitter, a micro-blogging website, has experienced tremendous growth in the last few years and has over 200 million active users who tweet over 400 million short-messages every day as of March 2013 [4]. Users post short-messages, most commonly known as, "tweet" which has a maximum of 140 characters. Tweet can contain text, emoticon (i.e. a combination of keyboard characters forms the facial expression. For example,:-) represents a smiling face), external link and hashtags. Users of social media tend to tweet using highly unstructured language with numerous typographical errors. A significant amount of tools and infrastructure is required to operationalize social media data due to its rapid growth and to the difficultly of processing its data by using standard relational SQL databases [5,6]. The typical change in the volume of data in social media related to a specific topic is often an indication of the occurrence of a significant event in the real world. Events in Twitter can be detected by clustering similar tweets.

The TDT system provided a general outline and fundamentals regarding event detection in earlier days. However, it is not a suitable solution for social media data as the TDT system is designed for topic detection over newswire. The text in social media is often noisy and the high volume of data presents challenges to the researchers attempting to find a technique suitable for event detection in social media. Today some solutions, including topic detection and tracking (TDT) [2,7,8] over newswire system and event detection in social media [9–11] have been employed in the data mining related researches. However, there is a great potential in research for extracting events from Twitter. We proposed a theoretical framework design for event detection and trending in multiple social networks presented in [12].

In this paper, the event detection and trending is conducted on a Twitter dataset which is divided into multiple chunks. Each chunk goes through the process of discovering the tweet clusters using LSH technique explained in Section 3.2 and their details are stored in a MySQL database. Each cluster centroid is identified by using *Frequency-based feature selection* explained in Section 3.5 and it is used to label the cluster. A novel methodology is used to find the interesting events by matching its keywords on cluster labels and trend them based on time, geo-locations and cluster size described in Algorithm-1. Also, the proposed methodology has a significant advantage that a number of clusters need not be supplied in advance as it leverages LSH technique to discover tweet clusters. Thus, it is an unsupervised model.

Experiments are conducted on Twitter dataset which contains the event "2011 End Times Prediction" [13–15]. The event was widely spread over popular news periodicals and social media websites. According to Camping's prediction, judgment day would take place on May 21st, 2011 and followers of Camping claimed that 3% of the world population would be raptured [13–15].

Finally, the clusters discovered using LSH and K-means are compared with clusters produced by hierarchical group average agglomerative clustering (GAAC). Section 4 discusses the results obtained from our experiments.

## 2. Related work

Online social media has attracted researchers due to its openness and the availability of data (e.g. Twitter access through TwitterAPI [16] and Facebook access through FacebookAPI [17]). Recent researches have made it evident that social media data captures vital information related to events in real-time. A large body of work pertaining to event detection has been carried out in the last few years. For example, an experiment by Bollen et al. demonstrated that stock market values can be predicted using public moods extracted from Twitter feeds [18]. A paper by Choi and Varian explained how search queries submitted to the Google search engine can be used to predict near-term values of economic indicators (For example, early prediction on automobile sales for a current month was predicted from *Google Trend*[1]) [19]. On other hand, Preis, Moat and Stanley used Google Trends data to predict the early warning signs of stock market fluctuations [20]. Yang et al. added two tasks in *Information Retrieval* (IR) for event detections: (1) retrospective detection and (2) online detection [8]. The former task detects events from accumulated data and the later finds events from news feeds in real-time. Yang et al. adapted a group average agglomerative clustering (GAAC) technique to group the documents in a collection. GAAC is a bottom-up hierarchical clustering technique, which starts clustering with each document as singleton cluster and agglomerate pairs of clusters based on their average similarity until all clusters merged into a single cluster that includes all documents [21]. Although the algorithm appears to be a natural solution for clustering, it has time and space complexity quadratic to the number of input documents [22]. Brants et al. adapted incremental TF–IDF and replaced cosine distance with Hellinger distance in their system to perform FSD in TDT datasets [23]. Parikh et al. developed an automatic scalable system for event detection from tweets, called ET system [24]. Key components of their approach were: (1) extraction of keywords to represent events, (2) an efficient storage mechanism to store the patterns, and (3) a hierarchical clustering technique for event detection. Sriram et al. proposed an approach to classify the tweets into one of several generic categories such as news, events, opinions, deals and private messages based on the author information and tweet features [25]. Tweet features used in the classification follows the categories' details, for example, token "deal" and special character like $ and % in tweet features follows the deals category) [25]. They also claimed that their approach outperforms the traditional bag of words (BOW) [21] strategy. Sankaranarayanan et al. built a system to discover news related tweets in Twitter [26]. However, a proper training set and collection of seeders are required for their approach, in order to construct a more generalized system. It would be very expensive to construct a training set for a volatile data stream. Document expansion for event detection based on agglomerative text clustering of semantically similar hashtags in Twitter has been experimented in [27]. A unified framework using hybrid clustering and topic model approach with rich features such as entities, time, and topics was proposed in [28]. However, both approaches do not explicitly improve the time complexity of the system which is an essential factor for high volume Twitter data stream. Becker et al. uses online clustering technique to group topically similar tweets and their system was able to classify them under real-world or non-real world events [29]. Packer et al. proposed an approach which associates tweets to a given event using query expansion [9].

Allan et al. presented the three TDT tasks of detection, first story detection and story link detection used by a system at University of Massachusetts (in short, it is referred as UMass) [2]. The k-nearest neighbor approach for the first story detection (FSD) used in UMass system declares a new document as a first story when its similarity to the closest document goes below a certain preset threshold. The nearest neighbor points are treated as similar items which are grouped to identify a story under a specific topic. The main problem of nearest neighbor is the representation of datasets in high *d*-dimensional vector space. The social media data is high dimensional in nature and needs to address the dimensionality reduction. The speed over linear search in k-nearest neighbor approach has significantly improved by an approximate nearest neighbor approach [30]. However, the indexing data structures proposed for the approximate nearest neighbor search, such as X-tree, R-tree and SR-tree, do not scale well with high dimension data [31]. Locality sensitive hashing (LSH), a technique for approximate nearest neighbor search, was first articulated in [2,7]. Since then, LSH has become a state-of-the-art technique for similarity search in high dimension data [11,30,32–34]. In text processing, LSH is capable of processing very large datasets of high dimensional items, which provides a high probability guarantee that it will return a correct or close match to the query (or search) document. LSH comprises two steps: (1) indexing the data and (2) searching for neighbors of a query point. Finding the nearest neighbors for the query point is computationally expensive when a given set of data is high-dimensional. Slaney et al. contributed an algorithm to calculate the optimum parameters for nearest-neighbor search using LSH [35]. Their work considers LSH based on p-distributions as introduced by Datar et al. [36] and distance between two points computed using the Euclidean $L_2$-norm function. Slaney et al. experimented the optimization parameters for LSH on wide variety of databases consist of images to achieve the expected results. Their work could be a good start for finding nearest neighbors in large datasets using randomized algorithm. The author concluded that it is difficult to find the best values for LSH parameters especially in large datasets. We explored a number of solutions to find appropriate set of hashing functions that are mathematically well-defined and fit the constraint of not colliding dissimilar items into the same bucket. Various experiments were conducted in this research work to compute the best values for LSH parameters. Similarly, in this work we found it is difficult to find the best values for LSH parameters. While different types of approaches have been carried by researchers toward event detection in social media, applying LSH technique for event detection in social media [11,37] is fairly a recent development. Petrovic et al. proposed a method using LSH to speedup FSD, which could run on social streams (i.e. on incoming tweets). In our proposed methodology, LSH approach is used for discovering tweet clusters to identify the events.

Monitoring events on social network gives insights into the events and also shows how trending changes over time. Some researchers have already shown the importance of finding the temporal pattern (e.g. emerging trend in local communities) in social media data stream for society's information and awareness [38,39]. Kwak et al. analyzed the tweets for top trending topics for a period of nearly four months and reported the temporal behavior of trends in terms of users' participation, active period of trend and number of tweets relevant to the topic [40]. A system for trend detection over Facebook public post was proposed in [41] where the characteristics of three different categories of trending topics such as disruptive events, popular topics and daily routines were analyzed. Twitter tracks the most popular phrases, words and hashtags and displays the top ten of each category under "trends" on a left side bar of user home page and allows users to customize them. However, Twitter does not provide details such as past trending topics, categories of trend item, user's participation and visualization of tweet geo-locations or trend's active period. In this paper, event trends are shown based on time, geo-locations and cluster size by analyzing the clusters and finally, the summary of the statistics for the event is reported.

## 3. Methodology

The processes involved in identification of events from tweet clusters which are discovered using LSH technique and also their trending are discussed in this section.

### 3.1. Tweet feature vector

Each tweet is fed into our system in the format of {tweet-id, user-id, geo-coordinates, timestamp, tweet}. The most meaningful representative terms or lexical items from a tweet are generally considered as features of that tweet. A tweet feature vector is a vector which mathematically encodes a set of features. Tweets are represented as vectors in $d$-dimensional vector space model [42], where $d$ is the number of different terms (i.e. features) appear in the tweets collection. Fig. 1 represents the system flow diagram for converting a raw tweet into the feature vector, which is later used as input for clustering technique. The creation of tweet feature vector is comprised of two sub-tasks: (1) qualifying a given tweet and (2) preparation of the feature vector for a tweet. A set of features are extracted from each qualified tweet. For simplicity, we considered only unigrams of tweet in our system. We convert each qualified tweet to its equivalent feature vector in second sub-task. Each sub-task is explained in more details below.

#### 3.1.1. Qualifying the tweets

A considerable level of noise in data is removed by this sub-task. Tokenizing is the process of splitting text into a set of individual terms. Each tweet is tokenized into a sequence of terms. Only tweets in English language are considered for clustering. A minimum of 30% of the words in a given tweet are required to be in English in order to qualify a tweet for further processing. The qualified tweet is checked against standard stopword list. The Natural Language Tool Kit's stopword list is used in this process to eliminate terms which carry the least information. For example, the articles like 'the', 'a', and 'an' are removed from the tweets. Terms contain only alphanumeric characters are treated as valid token. For the purpose of simplification, URLs and user names starting with '@' are removed and hashtags are retained in the filtration process. At end of this process, each tweet is split into a set of features which are included in vector space model.

To illustrate the above process, consider the following sample tweet:

LOVE THIS END OF THE WORLD SAGA IT IS JUST AMAZINGLY INTERESTING TO SEE PEOPLE FALL FOR IT. HOW INTERESTING http://bit.ly/dXBJyZ

After tokenization, language detection and filtration process, the following set of features are extracted:

{'love', 'end', 'world', 'saga', 'amazingly', 'people', 'fall'}

Though it is harder to read the pre-processed tweet, it reduces the number of dimensions in vector space model in compare to the original tweet. However, tweets in a moderate size dataset can still have thousands of dimensions.

#### 3.1.2. Incremental TF–IDF

For document vector space model, the collection of terms or features in dictionary defines the dimension of vector space where dictionary length is equal to number of dimensions. For dictionary term that does not appear in the document, the weight is considered as zero. Mathematically, a vector is quantified using its direction and magnitude. It can be a line of segment drawn from origin to a point in $d$-dimensional space. In *bag of words* [21] representation, the values of vectors are the number of occurrences of terms appear in the document. For example, a document {'winner', 'sports', 'winner'} corresponding to vector $(2, 1)$ has two unique terms 'sports' and 'winner', which define the dictionary that can be represented in two-dimensional vector space. The term 'winner' is associated with the $X$-axis and the term 'sports' is associated with the $Y$-axis. A line can be drawn from $(0, 0)$ to $(2, 1)$ represent the document vector in two-dimensional vector space model. The

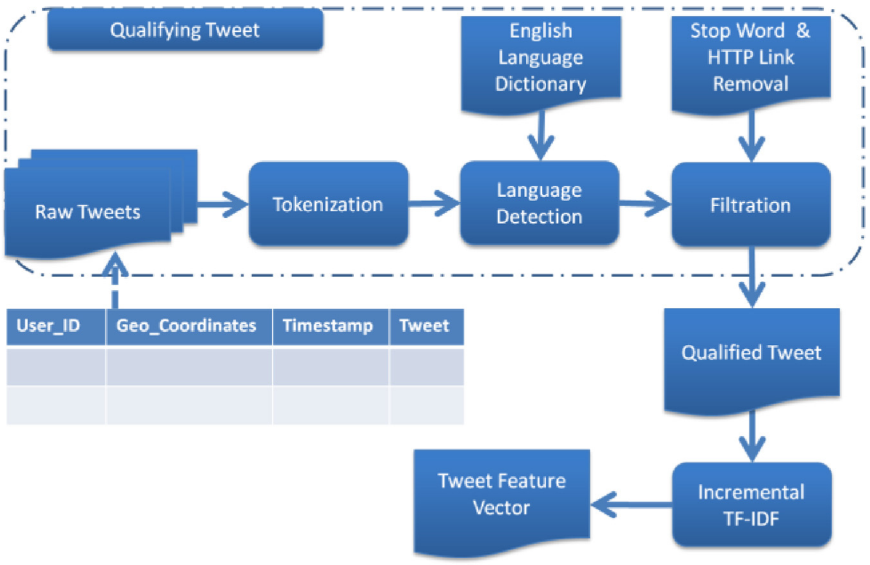

**Fig. 1.** System flow for creation of Tweet feature vector.

same idea can be extended to any number of terms in documents. In practice, there are several term weighing schemes have been used to assign values or magnitude for terms in a document vector. Term frequency–inverse document frequency (TF–IDF) is one of the most fundamental techniques in IR which is a product of term frequency (TF) and inverse document frequency (IDF) where TF represents the importance of the term in a document and IDF represents the importance of term in entire corpus [21,42]. Let us consider a document $d$ has set of terms. TF–IDF weighing scheme assigns weight to term '$w$' in document '$d$' using the following formula [43]:

$$tf\text{–}idf(w,d) = tf(w,d) * idf(w) \tag{1}$$

$$idf(w) = \log \frac{N}{df(w)} \tag{2}$$

where $tf\text{–}idf(w,d)$ is known as TF–IDF weight. $tf(w,d)$ is a term frequency which represents the number of times the term $w$ occurs in a document $d$. $idf(w)$ is an inverse document frequency for the term $w$ where $N$ is the number of documents in a data corpus and $df(w)$ is the number of documents in which the term $w$ occurs atleast once.

Today social networking sites produce large size of data corpus and it is difficult to fit them using document vector space model due to the limitation of memory. Updating TF–IDF weight of each term in a dictionary is a challenging task due to the growing Twitter data corpus. There could be two possible ways to handle this problem. Either we could use a static dictionary build from fixed size data corpus or incrementally update the term weight every time the new document arrives which is known as incremental TF–IDF [8,23]. A combination of these two approaches is used in this paper. To the best of our knowledge, it is a first time that these approaches are combinedly used in social media data to solve the aforementioned problem. The data corpus is divided into chunks of equal size where each chunk has no more than '$N$' tweets. Let us consider the data corpus $D = \{D_1, D_2, D_3, \ldots, D_n\}$ where $D_1, D_2, D_3, \ldots, D_n$ are tweet chunks. The static dictionary is constructed for each chunk $D_i$ based on incremental TF–IDF. For better utilization of memory, TF–IDF weight is updated for each term in the dictionary by processing each tweet one at a time in $D_i$. Hence, Eqs. (1) and (2) are rewritten and the incremental TF–IDF is defined is below:

$$tf\text{–}idf_i(w,d) = tf(w,d) * idf(w) \tag{3}$$

$$idf(w) = \log \frac{N_i}{df_i(w)} \tag{4}$$

where $tf\text{–}idf_i(w,d)$ is $tf\text{–}idf$ for term $w$ in tweet $d$ from tweet chunk $D_i$. $idf(w)$ is an inverse document frequency for the term $w$ where $N_i$ is the number of documents in $D_i$ and $df_i(w)$ is the number of documents from $D_i$ in which the term $w$ occurs atleast once. Finally, tweet features are mapped to the dictionary to generate its tweet feature vector. In summary, a static dictionary for each tweet chunk is constructed using incremental TF–IDF described above in order to represent a tweet vector in high-dimensional vector space.

A similarity measure is a function that computes the degree of similarity between two documents which is essential while clustering similar documents. Let us consider two document vectors $\vec{v}(d1)$ and $\vec{v}(d2)$ derived from documents $d1$ and $d2$ respectively. Quantifying the similarity measure between them using its magnitude yield a significant vector difference when both documents have similar content with different vector size. The standard way to solve this issue is by using cosine similarity which is a dot product of two vectors and is defined below [43]:

$$\begin{aligned} sim(d1,d2) &= \cos(d1,d2) = \frac{\vec{v}(d1)\cdot\vec{v}(2)}{|\vec{v}(d1)||\vec{v}(d2)|} \\ &= \frac{\sum_{i=1}^{M} tf\text{–}idf_i(w,d_1) * tf\text{–}idf_i(w,d_2)}{\sqrt{\sum_{i=1}^{M} tf\text{–}idf_i^2(w,d_1)} * \sqrt{\sum_{i=1}^{M} tf\text{–}idf_i^2(w,d_2)}} \\ &= \vec{v}(d1)\cdot\vec{v}(2) \end{aligned} \tag{5}$$

where the numerator is the dot product of two vectors and the denominator is the product of Euclidean length of them.

From Eq. (5), the Euclidean length of any vector document $\vec{v}(d)$ with $M$ terms is defined as below [43]:

$$|\vec{V}(d)| = \sqrt{\sum_{i=1}^{M} tf\text{–}idf_i^2(w,d)} \tag{6}$$

and the document vector can be length-normalized to the unit vector as follows [43]:

$$\vec{V}(d) = \frac{\vec{V}(d)}{|\vec{V}(d)|} \tag{7}$$

In this paper, the similarities between documents are calculated using cosine similarity which is defined in Eq. (5).

### 3.2. Locality sensitive hashing

Event detection is a research topic initiated by Allan through his TDT project [7] and the objective is to identify the topics in news streams such as newswire, radio broadcasts and TV news shows. However, the system does not provide a feasible solution to handle rapidly growing social media data stream as it was built mainly for news streams. The hashing function for finding similar documents was first introduced by Broder [44]. Indyk and Motwani used locality sensitive hashing (LSH) technique, the most relaxed version of nearest neighbor called "approximate nearest neighbor approach" to detect unseen stories in a stream of broadcast news stories [30]. It enables the fast and approximate comparisons of vectors using cosine similarity. The key idea of LSH is to apply the hash functions in such a manner that the probability of collision is much higher for similar documents than for documents that are dissimilar [30]. In other words, objects close to each other will most likely fall into the same bucket. Formally, the LSH is defined as follows [30]:

**Definition.** Let us consider a ball of radius $r$ for distance measure $D$ as $\text{Ball}(q,r)=\{p: D(q,r)\geq r\}$ and $H$ is a family of hash functions which maps the given space $S$ to $U$ for any randomly selected hash function $h_{r1,r2,\ldots,rk}$ (i.e. $H=\{h: S\to U\}$). A family of functions $H$ is said to be $(r_1,r_2,p_1,p_2)$ – sensitive if any pair of data points $p, q \in S$ satisfies the following properties:

- if $p\in \text{Ball}(q,r_1)$ then $\Pr_H[h(q)=h(p)]\geq p_1$,
- if $p\notin \text{Ball}(q,r_2)$ then $\Pr_H[h(q)=h(p)]\leq p_2$.

where $\Pr_H[h(q)=h(p)]$ is the probability that the value of hash function on two data point $p$ and $q$ returning the same value.

An overall idea of LSH algorithm is visually represented in Fig. 2 [45]. Let us consider $N$ is a set of data points in '$S$' space. The data points are preprocessed using the randomly chosen independent hash function $h_{r1,r2,\ldots,rk}$ and stored in a hash table. The hash values of the data points act as index of the hash table. For any query point $q$, the hash function $h_{r1,r2,\ldots,rk}$ is applied to determine the index of the hash table from which the set of closely related data points are fetched. From the search result, the most similar '$n$' neighbors can be fetched and as a result, they are called as nearest neighbor of the query data point $q$.

Charikar used LSH in [46] to map high dimensional vectors to low dimensional space while preserving similarity between vectors in original space. LSH algorithm computes a signature of feature vectors by computing dot product of feature vector $\vec{u}$ with random unit vector $\vec{r}$ drawn from $d$-dimensional Gaussian distribution of value $N(0,1)$ and retaining sign of resulting product as follows [46]:

$$h_{\vec{r}}(\vec{u}) = \begin{Bmatrix} 1 & \text{if}\,\vec{r}\cdot\vec{u}\geq 0 \\ 0 & \text{if}\,\vec{r}\cdot\vec{u}<0 \end{Bmatrix} \quad (8)$$

where $h_{\vec{r}}$ is a hash function which can used to find the hamming distance between feature vectors. As an example, a 13-bit tweet signature for a sample tweet that is shown below is found using Eq. (8).

Tweet:

LOVE THIS END OF THE WORLD SAGA IT IS JUST AMAZINGLY INTERESTING TO SEE PEOPLE FALL FOR IT. HOW INTERESTING http://bit.ly/dXBJyZ

Sigature: bitarray('0011101000100')

Charikar's LSH scheme for similarity measure is measured by cosine angle distance between two vectors and it is proportional to the expected hamming distance of their signature vectors while preserving the cosine similarity in high dimensional space. Thus, for a given pair of vectors, the cosine similarity of them is defined in (5) as the dot product of feature vectors normalized by their length. Ravichandran et al. made use of LSH algorithm for searching similar nouns from large web corpus [47].

There are three significant parameters in LSH which needs to be defined prior to document clustering: (1) number of permutations ($P$), (2) signature length ($K$) and (3) probability of finding nearest neighbor with a cosine similarity ($T$). Experiments are conducted for finding values for those parameters and are explained in Section 4.2. In our system, the nearest neighbor search in LSH is sped up by using prefix tree data structure explained in Section 3.3 and events are detected from tweet clusters explained in Section 3.4.

### 3.3. Finding nearest neighbor of tweet using prefix tree

A prefix tree or trie is a tree-like data structure to store strings for fast pattern matching. Trie is commonly pronounced as "tree" or "trie" routed from the word "re$\underline{\text{trie}}$val" [48]. Trie data structure associates a key with value and was created by Fredkin [49]. Tries can be used to perform query operations for pattern matching and prefix matching. Therefore, it is also known as prefix tree which can be used for information retrieval application where it is required to store key and value pair information and retrieve them quickly. Prefix tree or tries do not have collision. Hence, we leverage the prefix tree data structure to find the nearest neighbor of a given tweet. The procedure of finding the nearest neighbor of a user described by Kamath and Caverlee [50] is employed in this research work to find the nearest neighbor of tweet. Let us assume the nearest neighbor of a tweet $\vec{t}$ is $\vec{t}_n$, which belongs to cluster $C_n$ and a set of permutation functions $H=\{\pi_1, \pi_2, \pi_3, \ldots, \pi_P\}$, with each permutation function in the following form [47]:

$$\pi(X) = (aX+b) \bmod p \quad (9)$$

where $a$ and $b$ are chosen randomly and $p$ is a prime number. The collection of $P$ number of prefix tree is defined where every prefix tree corresponds to a permutation function in $H$. Hence, $P$-times a tweet is permutated and they are stored in $P$.

To compute the nearest neighbor and its cluster for the given tweet, first the signature of the given tweet is calculated. Then, $P$-times the signature is permutated and the nearest neighbor is found in prefix tree by iterating the tree starting from its root. At the end of the step, possibly $P$ number of neighbors with their respective clusters are detected for the given tweet. The nearest neighbor is the one which has minimal hamming distance. The tweet is added into the cluster of the nearest neighbor. Prefix trees are used in the place of buckets in LSH to speed up the process. If there is no nearest neighbor for a tweet, a new cluster is created with the given tweet.

### 3.4. Event detection and trending

The procedure to find the events from tweet clusters is described in this section. Algorithm-1 shows the high level process for event detection and trending from tweet clusters discovered using prefix-tree based LSH. It is a modified version of algorithm used by Kamath and Caverlee [50] in which the purpose of clustering is to identify newly formed communities of users from the real-time web. Let us consider the Twitter data corpus $D=\{D_1, D_2, D_3, \ldots, D_n\}$ as discussed in Section 3.1.2, where $|D_I|\leq N$ and $D_I=\{t_1, t_2, t_3, \ldots, t_i\}$ for each $D_I\in D$. $\{t_1, t_2, t_3, \ldots, t_i\}$ is a set of tweets in a chunk $D_I$.

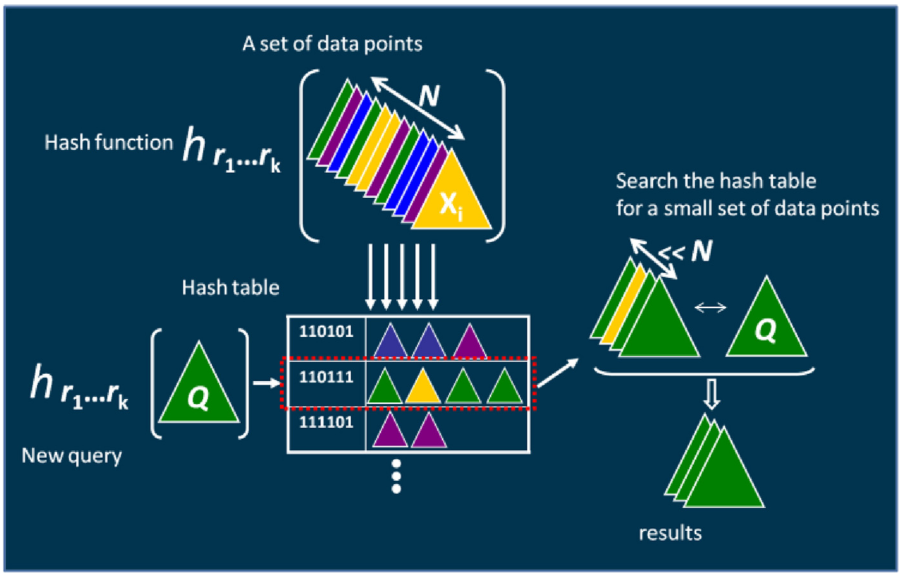

**Fig. 2.** LSH with random projection [45].

***Algorithm-1***: Event Detection and Trending from Tweet Clusters Discovered by LSH

*Given*: set of tweets $D_I = \{t_1, t_2, t_3, \ldots t_i\}$ where each tweet in the format of {tweet-id, user-id, geo- coordinates, timestamp, tweet} , Cluster Setting { number of Permutations P, signature length K, probability of finding nearest neighbour with a cosine similarity T}, keywordsInEvent
*Output*: set of clusters $C_R = \{C_1, C_2, C_3, \ldots C_r\}$, event summary and event trends

```
1.  Build dictionary $Dict_I$ for $D_I$ with length m using incremental TF-IDF
2.  Create a set of random vectors in Gaussian distribution R
        /* Initialize prefix trees using permutation functions $\mathcal{H}$ as described in section 3.3 */
3.  for each i in 1 to P
4.      $\pi_i \sim \mathcal{H}$
5.      Create prefixtree $P_{tree}[i]$
6.  end for
7.  for each tweet $t \in D_I$ do:
            /* Construct tweet vector using $Dict_I$ as described in section 3.1 */
8.      Construct tweet vector using $Dict_I$
            /* Create K-bit signature for tweet vector as described in section 3.2 */
9.      Create K-bit tweet signature using R
            /* Insert K-bit tweet signature into P number of Prefix Trees */
10.     for each i in 1 to P
11.         Insert tweet signature Into prefextree $P_{tree}[i]\ [\pi_i(t)]$
12.     end for
        /* Get the nearest neighbour $t_n$ and its cluster $C_n$ of tweet as described in section 3.3 */
13.     Generate new tweet signature using R
14.     $t_n \leftarrow$ get neartest neighbour of tweet from prefextree $P_{tree}[i]\ [\pi_i(t)]$
15.     if sim( t , $t_n$) $\geq$ T then
16.         if t is not in cluster $C_n$ then:
17.             add t to cluster $C_n$
18.         end if
19.     else
20.         create new cluster C with t
21.     end if
22. end for
```

```
            /* |c| ≠ 1 ignores singleton cluster */
23. for each cluster c ∈ C and |c| ≠ 1 do:
            /* GetClusterLabel uses frequency-based frequency selection described in Section 3.5 for
            finding the cluster label */
24.     storeClusterToDB(cluster-id,GetClusterLabel(c), cluster-size)
25.     for each t in c do:
26.         storeTweetToDB(cluster-id,tweet-id,geo-coordinates,user-id,timestamp,tweetvector)
27.     end for
28. end for
29. for each cluster c fetch from DB do:
30.     select top n-terms 'l' from cluster_label of c
31.     If l match with KeywordsInEvent then: add c to E
32. end for
33. Consolidate clusters in E and generate event summary (number-of-tweets,number-of-unique-users, tag-cloud)
34. Generate event trend based on time, geo-location, cluster size from tweets of E
```

It is necessary that the tweets in Twitter data corpus *D* span over a specific time interval. Our experiment covers one day of tweets from the dataset, as described in Section 4.1. Algorithm-1 is used for all chunks in *D* to produce the tweet clusters. After processing all tweets in *D*, the clusters are analyzed further for the presence of events and the event trends are generated. A cluster with a single tweet is ignored in cluster analysis due to the fact that its single tweet cannot form the event. The algorithm is evaluated on set of tweets posted on Twitter in a specific time interval and the experiment is discussed further in Section 4.

## 3.5. System architecture

Fig. 3 provides an architectural overview of the proposed system for event detection and trending. The Twitter data corpus is divided into the chunks of equal size. Each chunk contains no more than *n* number of tweets and is processed individually to find the clusters. The cluster detail (i.e. cluster label and information of individual tweets within that cluster) is stored in a MySQL database which is used later to generate the event trends. The steps involved in event detection and trending are explained below:

*Step 1*: Each tweet in a chunk is qualified and is converted into a tweet feature vector mapped with the dictionary as described in Section 3.1.
*Step 2*: In order to reduce the dimension of the input vector, each tweet feature vector is further converted into its equivalent *k*-bit signature using a method proposed by Charikar described in Section 3.2.

*Step 3*: The *k*-bit tweet signature received from Step 2 is added to its respective cluster using algorithm-1 (10–21) described in Section 3.4. Steps 1–3 are repeated for all tweets in a chunk. A set of clusters are discovered at the end of Step 3.

Selection of the most frequently occurring terms in a cluster is called *Frequency-based feature selection* method which can be used to label the cluster as described in [21]. In this paper, this method is applied to select the list of frequently occurring terms from each cluster. The most frequent terms are considered as cluster centroid, and are used to label the cluster. The cluster attributes such as cluster label containing most frequently used unigrams and their weights (including hashtags) and details of each tweet {user-id, geo-coordinates, timestamp} that appear in cluster are stored in the database. The minimum information for a cluster is stored in the database for effective use of storage.

The above procedure is conducted for all tweet chunks. The dictionary is flushed and reconstructed every time a new chunk is processed.
*Step 4*: Cluster analysis is performed after processing all tweets from a data corpus. Cluster attributes are fetched from the database and analyzed independently. Key search is made on cluster labels to find the matching clusters for an event.
*Step 5*: Behavior of the event is shown in terms of its trend using the information provided by consolidated clusters. The event is trended base on time, geo-locations and cluster size. Algorithm-1 (23–34) in Section 3.4 describes the Steps-4 and 5.

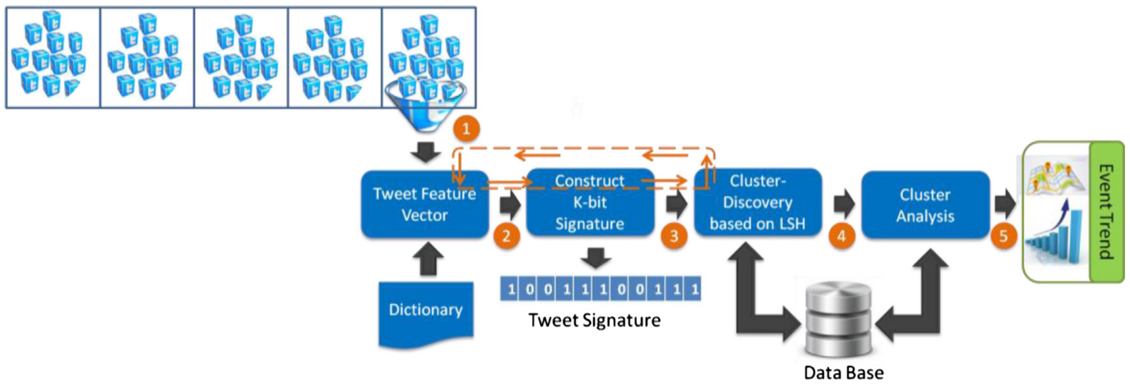

**Fig. 3.** System architecture for event detection and trending for tweets.

## 4. Experiments

This section discusses the experiment conducted for event detection and trending using tweet clusters. Cluster quality compares how well different techniques perform clustering on a given dataset. In this paper, the quality of clusters discovered by LSH and K-means are compared to hierarchical GAAC technique. Different values for each LSH parameter influencing the cluster quality are examined and the best suitable values are reported. Finally, the clusters are analyzed for the presence of an event and its trends are shown in the next sections.

### *4.1. Dataset*

The proposed methodology is investigated on Twitter dataset containing the event "2011 End Times Prediction" [13–15], hereafter; referred to as "EndOfWorld". One million tweets of four days between 18-May-2011 and 21-May-2011 are extracted from Twitter from which the tweets related to keyword "End%of%World" are queried. The html of retrieved tweet is fragmented into the required format: {tweet-id, user-id, geo-coordinates, timestamp, tweet}. The qualified tweets are extracted by using the method explained in Section 3.1.1. Fig. 4 shows four days of tweets related to "EndofWorld" event. Fig. 5 shows the linear growth in dimension of unique tokens found from one day (i.e. tweets posted on May 21, 2011) Twitter dataset containing 1694 tweets. Initially the Porter stemmer [21] is used to remove the common morphological endings of each token. For example, Porter changes the term "discovering" to "discover". The Porter stemmer does not show any significant impact when the good cluster technique is applied [51]. According the experimental result shown in Fig. 5, it is observed that the dimensions have not been significantly reduced. Thus, stemming is not engaged while preparing the tweet feature vector. Charikar's family of hash functions is applied to reduce the high dimensional vectors into *k*-bit signature array of much smaller dimension. Finally, Algorithm-1 is applied for event detection and trending on one day (May 21, 2011) Twitter dataset containing 1694 tweets.

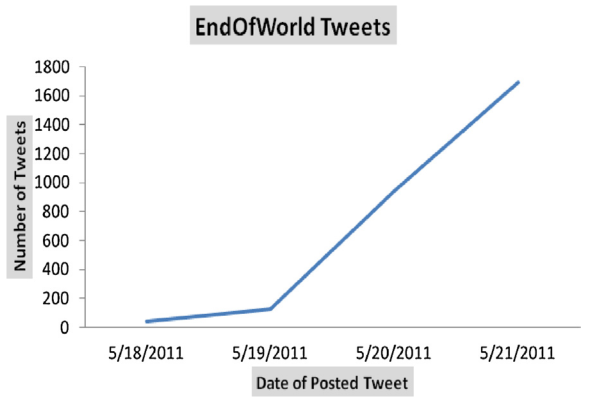

**Fig. 4.** Four days of EndofWorld tweets.

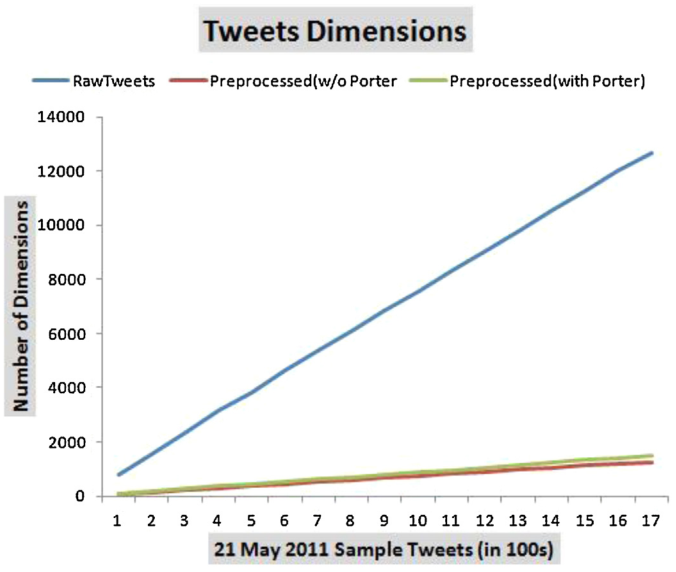

**Fig. 5.** Linear growth of tweets' dimensions.

### *4.2. Evaluation of cluster-discovery techniques*

Document needs to be pre-categorized in order to evaluate the clusters produced by different techniques. This can be achieved by human labeling or automatic labeling. Human labeling would be extremely time consuming process especially on large social media datasets. Automatic labeling would be economically effective as it utilizes the less time which can be checked by domain experts. Hierarchical agglomerative clustering technique has recently been used to automatically categorize the documents in [52]. GAAC clustering of documents in one language (English) are used as "gold standard" and they are compared with clusters produced for another language (Bulgarian) in [52]. In their experiments, cosine distance is used to compute the similarity between two documents. The number of clusters in given data set is usually unreliable since the nature of data is unknown. However, value for number of clusters is retrieved by processing the documents using LSH and then, it is used as a parameter for GAAC and K-means.

The quality of the clusters discovered by LSH and K-means are compared against GAAC. Purity and normalized mutual information (NMI) are external evaluation criterion for cluster quality [21]. GAAC technique is considered as a golden truth clustering class in cluster evaluation. Let us consider the results of clusters $K = \{C_1, C_2, C_3, \ldots, C_n\}$ and the golden truth of clusters $\Omega = \{W_1, W_2, W_3, \ldots, W_w\}$for '*N*' number of tweets. Tweets in predicted clusters are labeled according to GAAC and quality is measured as follows:

*Purity:* To calculate the purity, each cluster is assigned to the class, which is most frequent in the cluster and then, the purity of the cluster is measured as below [21]:

$$\text{Purity}\quad(K, \Omega) = \frac{\text{Number of correctly assigned documents}}{\text{Total number of documents}} \tag{10}$$

Formally,

$$\text{Purity}\quad(K, \Omega) = \frac{1}{N}\sum_{N} \max_w |C_n \cap W_w| \tag{11}$$

*Normalized mutual information (NMI):* High purity can be achieved when the number of clusters is large in which each document gets its own cluster. Thus, we use normalized mutual information (or NMI) measure to avoid a tradeoff between the quality and the number of clusters. NMI is a normalized mutual information score ranges between 0 (no mutual information) and 1 (perfect correlation) and it is computed using mutual information (MI) and entropy.

Entropy is a measure of uncertainty for a partition set. For results clusters ($K$) and golden truth clusters ($\Omega$), their entropy is defined by [21] as follows:

$$H(K) = -\sum_{n} \frac{|C_n|}{N} \log \frac{|C_n|}{N} \tag{12}$$

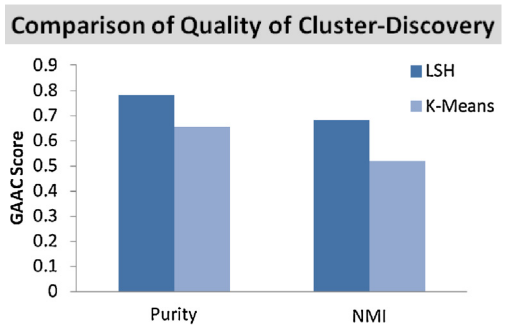

**Fig. 6.** Quality of clusters.

where $|C_n|/N$ is the probability that a tweet picked at random falls into a cluster $|C_n|$, likewise for $\Omega$:

$$H(\Omega) = -\sum_{w} \frac{|W_w|}{N} \log \frac{|W_w|}{N} \tag{13}$$

The mutual information (MI) between $K$ and $\Omega$ is calculated by [21]:

$$\mathrm{MI}(K, \Omega) = \sum_{n}\sum_{w} \frac{|C_n \cap W_w|}{N} \log \frac{N|C_n \cap W_w|}{|C_n||W_w|} \tag{14}$$

where $|C_n \cap W_w|/N$ is the probability that a tweet picked at random falls into both $C_n$ and $W_w$.

Finally, NMI is formally defined as [21]:

$$\mathrm{NMI}(K, \Omega) = \frac{\mathrm{MI}(K, \Omega)}{[H(K) + H(\Omega)]/2} \tag{15}$$

Fig. 6 shows the comparison between the quality of clusters discovered using K-means and LSH against hierarchical GAAC. From the experiments, it is noticed that LSH attained 12.5% and 16.6% of purity and NMI respectively more than K-means. Fig. 7 shows the running time required for the proposed LSH scheme and K-means algorithm to discover clusters from a given dataset. LSH took only 12.99% of average running time required for K-means and was consistently about 0.3 s during the complete cluster discovery process. It is also observed that there is a significant improvement in running time required for LSH compared to K-means while maintaining the quality of clusters. Experiments are repeated with varying the choice of LSH parameters: number of permutations $P$, signature

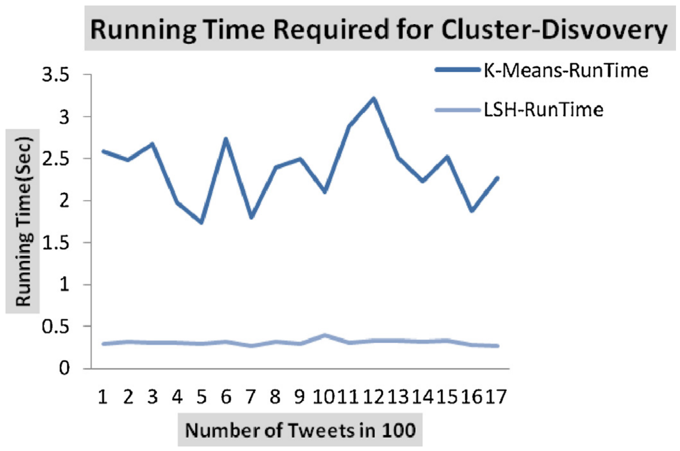

**Fig. 7.** Running time required for LSH and K-means.

length $K$ and the probability of finding nearest neighbor with a cosine similarity $T$.

The experimental result of the effect of LSH parameters on purity and NMI with $T = 0.005$ reported in Table 1. We also conducted the experiments by setting different values for $T$ with an interval of 0.1 in the range of 0.005–0.605. During experiments, it is observed that the increase in the value of $T$ identifies a large number of singleton clusters (i.e. the cluster with a single tweet). For the given dataset, $T = 0.005$ yielded a better result in terms of NMI and henceforth, the experiment uses LSH values $P = 23$, $K = 17$ and $T = 0.005$ for cluster discovery process.

### 4.3. Cluster analysis for event trending

We analyzed each cluster retrieved using LSH scheme. We applied frequency-based feature selection to build the list of frequently occurring terms for each cluster. The most frequent terms are considered as cluster centroid, and are used to label the cluster. Fig. 8 shows the size of each cluster related to the event spanning over a time period of 1 day (May 21, 2011). These clusters were consolidated for trending purposes. The behavior of the event at each hour based on the number of tweets is shown in Fig. 9 where it is noticed that the maximum number of tweets was posted between 16th and 17th hour of that day. It was predicted that the world will end on May 21, 2011 at 6 PM. According to Fig. 9, we can see that

**Table 1**
Effect of LSH parameters ($P$ and $K$) on purity and NMI for $T = 0.005$.

| $K$ | 13 | | 17 | | 19 | |
|---|---|---|---|---|---|---|
| $P$ | Purity | NMI | Purity | NMI | Purity | NMI |
| 11 | 0.677083 | 0.5542136 | 0.73958 | 0.607169 | 0.73958 | 0.568426 |
| 13 | 0.708333 | 0.5707411 | 0.76042 | 0.623468 | 0.6875 | 0.55999 |
| 17 | 0.7708 | 0.63436 | 0.80208 | 0.617016 | 0.72917 | 0.57377 |
| 19 | 0.75 | 0.5730382 | 0.6875 | 0.575469 | 0.76042 | 0.646738 |
| 23 | 0.6875 | 0.5374632 | **0.7813** | **0.68302** | 0.77083 | 0.611772 |
| 29 | 0.739583 | 0.6320713 | 0.78125 | 0.631359 | 0.77083 | 0.623051 |
| 31 | 0.739583 | 0.5960132 | 0.76042 | 0.602494 | 0.78125 | 0.629202 |
| 37 | 0.729167 | 0.5843418 | 0.77083 | 0.611772 | 0.77083 | 0.611772 |
| 41 | 0.75 | 0.6179423 | 0.71875 | 0.543036 | 0.7917 | 0.6553 |
| 43 | 0.770833 | 0.6332873 | 0.77083 | 0.611772 | 0.71875 | 0.576992 |
| 47 | 0.739583 | 0.5816589 | 0.78125 | 0.63826 | 0.76042 | 0.583202 |
| 53 | 0.729167 | 0.6024508 | 0.77083 | 0.611772 | 0.77083 | 0.611772 |
| 59 | 0.71875 | 0.5828351 | 0.8125 | 0.637015 | 0.72917 | 0.564439 |
| 61 | 0.739583 | 0.5909974 | 0.77083 | 0.611772 | 0.76042 | 0.578294 |
| 67 | 0.770833 | 0.611772 | 0.79167 | 0.628863 | 0.77083 | 0.611772 |
| 71 | 0.770833 | 0.6131527 | 0.77083 | 0.611772 | 0.77083 | 0.611772 |
| 73 | 0.75 | 0.58081 | 0.75 | 0.610804 | 0.72917 | 0.564439 |
| 79 | 0.770833 | 0.6214451 | 0.77083 | 0.618078 | 0.76042 | 0.59378 |

Purity and NMI values corresponding to $K = 17$ and $P = 23$ which have been used for the experiments.

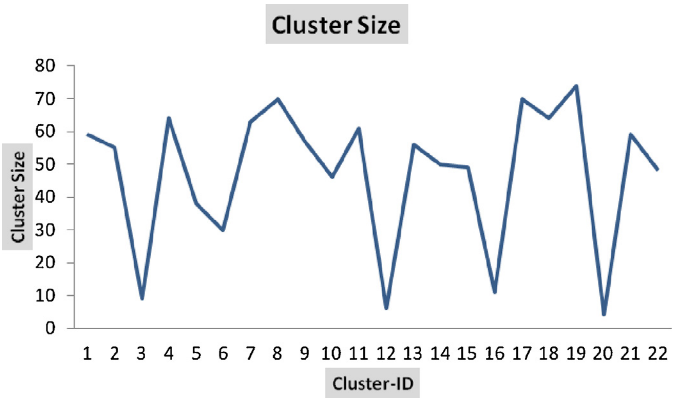

**Fig. 8.** Size of each cluster found in the sample tweets (time span = 1 day).

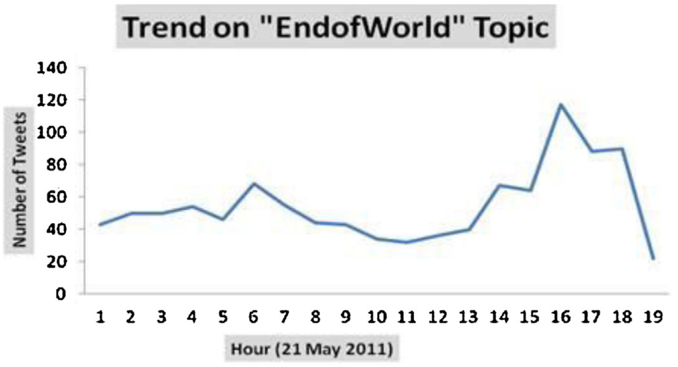

**Fig. 9.** 21 May 2011 tweets trending on "EndofWorld".

the frequency of tweets is increasing until the 16th hour of the day after which it is started to diminish. This type of trending graph illustrates the popularity of the event over time. Fig. 10 shows the Google map for geographically distributed tweets of consolidated clusters from which it is noticed that users from different part of the world discussed about the event and the majority of the tweets are from Northern America. The event summary such as number of unique users and number of tweets of consolidated cluster is reported in Fig. 11. A tag cloud or word cloud is a visual representation for text data which gives greater prominence to words that appear more frequently in data source. In this paper, a set of highly weighted terms (i.e. most frequently appearing terms) of

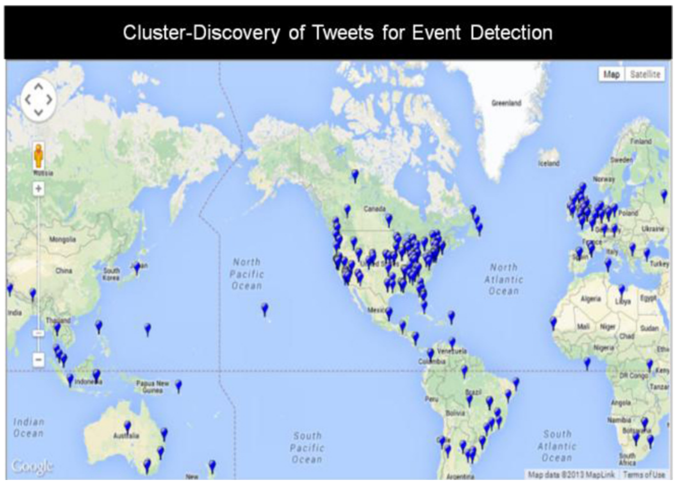

**Fig. 10.** Tweets on "EndofWorld" Event on Google Map.

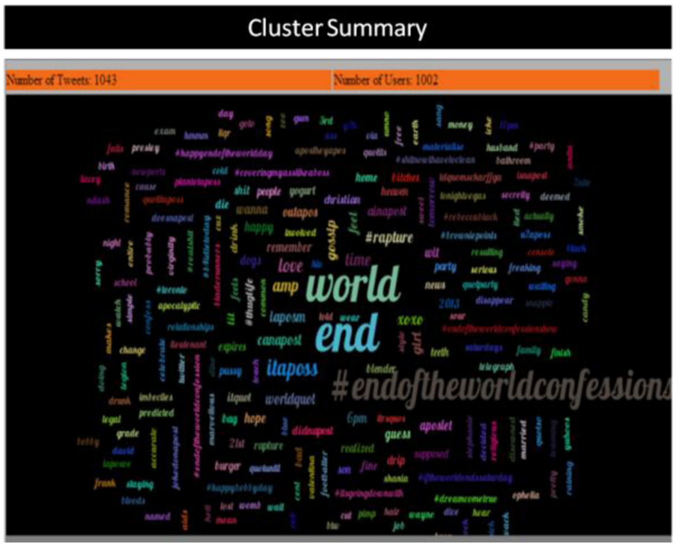

**Fig. 11.** Cluster summary.

consolidated cluster is used to form a tag cloud. The importance of words is shown with font size. As shown in Fig. 11, the keywords 'world', 'end', '#endoftheworldconfession', 'rapture', etc., are frequently appearing unigrams which related the event "EndOfWorld". The identified event is cross checked with Wikipedia to ensure that it is not a spam and is an existing topic.

## 5. Conclusion and future work

We presented a novel methodology described in Algorithm-1 to find an interesting event by matching its keywords on cluster labels and trend it based on time, geo-locations and cluster size. The problem of creating the feature vector in high dimensional data is solved by using a static dictionary constructed using an incremental TF–IDF technique.

Since the proposed methodology leverages the LSH technique to find the tweet clusters, it has a significant advantage that a number of clusters need not be supplied in advance. Runtime required for cluster discovery process is significantly improved by using prefix tree data structure in LSH technique. It is also observed that LSH based cluster-discovery algorithm outperforms K-means while maintaining the quality of clusters.

In our future work, we are interested in applying our methodology on multiple social media dataset to see the effect of same events across different social media platforms, and the proposed methodology can be extended to develop a more sophisticated trending system to generate event trends for user queried keywords.

## Acknowledgements

The authors would like to express their gratitude to the Natural Sciences and Engineering Research Council of Canada and Ryerson University, Faculty of Science for the financial support of this research.

## References

[1] M.R. Morris, J. Teevan, K. Panovich, What do people ask their social networks, and why? A survey study of status message and behavior, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, GA, USA, 2010, pp. 1739–1748.

[2] J. Allan, V. Lavrenko, D. Malin, R. Swan, Detections, bounds, and timelines: UMass and TDT-3, in: Proceedings of Topic Detection and Tracking Workshop, 2000, pp. 167–174.

[3] J.G. Fiscus, G.R. Doddington, in: J. Allan (Ed.), Topic Detection and Tracking, Kluwer Academic Publishers, Norwell, MA, USA, 2002, pp. 17–31.
[4] W. Karen, Celebrating #Twitter7, 2013, March, Available at: https://blog.twitter.com/2013/celebrating-twitter7
[5] J. Lin, D. Ryaboy, Scaling big data mining infrastructure: the Twitter experience, SIGKDD Explor. Newsl. 14 (April) (2013) 6–19.
[6] G. Mishne, J. Dalton, Z. Li, A. Sharma, J. Lin, Fast data in the era of big data: Twitter's real-time related query suggestion architecture, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 2013, pp. 1147–1158.
[7] J. Allan, in: J. Allan (Ed.), Topic Detection and Tracking, Kluwer Academic Publishers, Norwell, MA, USA, 2002, pp. 1–16.
[8] Y. Yang, T. Pierce, J. Carbonell, A study of retrospective and on-line event detection, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 1998, pp. 28–36.
[9] H.S. Packer, S. Samangooei, J.S. Hare, N. Gibbins, P.H. Lewis, Event detection using Twitter and structured semantic query expansion, in: Proceedings of the 1st International Workshop on Multimodal Crowd Sensing, Maui, HI, USA, 2012, pp. 7–14.
[10] T. Sakaki, M. Okazaki, Y. Matsuo, Earthquake shakes Twitter users: real-time event detection by social sensors, in: Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 2010, pp. 851–860.
[11] S. Petrovic, M. Osborne, V. Lavrenko, Streaming first story detection with application to Twitter, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, CA, 2010, pp. 181–189.
[12] S.B. Kaleel, M. AlMeshary, A. Abhari, Event detection and trending in multiple social networking sites, in: Proceedings of the 16th Communications & Networking Symposium, San Diego, CA, 2013, pp. 5:1–5:5.
[13] A conversation with Harold Camping, Prophesier of Judgment Day, New York Magazine May (2011).
[14] C. Goffard, Harold Camping is at the Heart of a Mediapocalypse, 2011, May, Available at: http://articles.latimes.com/2011/may/21/local/la-me-rapture-20110521
[15] Family Radio, The end of the world is here! Holy god will bring judgement day on May 21, 2011, 2011, May, Available: http://web.archive.org/web/20110608223300/http://www.familyradio.com/graphical/literature/judgment/judgment.html
[16] Twitter, REST API v1.1 Resources, 2014, Available at: https://dev.twitter.com/docs/api/1.1
[17] Facebook, Quickstart for Graph API, 2014, Available at: https://developers.facebook.com/docs/graph-api/quickstart/
[18] J. Bollen, H. Mao, X. Zeng, Twitter mood predicts the stock market, J. Comput. Sci. 2 (2011) 1–8.
[19] H. Choi, H. Varian, Predicting the present with Google trends, Econ. Rec. 88 (2012) 2–9.
[20] T. Preis, H.S. Moat, H.E. Stanley, Quantifying trading behavior in financial markets using Google Trends, Sci. Rep. 3 (2013).
[21] C.D. Manning, P. Raghavan, H. Schutze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.
[22] D.R. Cutting, D.R. Karger, J.O. Pedersen, J.W. Tukey, Scatter/gather: a cluster-based approach to browsing large document collections, in: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, 1992, pp. 318–329.
[23] T. Brants, F. Chen, A. Farahat, A system for new event detection, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, 2003, pp. 330–337.
[24] R. Parikh, K. Karlapalem, ET: events from tweets, in: Proceedings of the 22nd International Conference on World Wide Web Companion, Rio de Janeiro, Brazil, 2013, pp. 613–620.
[25] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, M. Demirbas, Short text classification in Twitter to improve information filtering, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Geneva, Switzerland, 2010, pp. 841–842.
[26] J. Sankaranarayanan, H. Samet, B.E. Teitler, M.D. Lieberman, J. Sperling, TwitterStand: news in tweets, in: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, 2009, pp. 42–51.
[27] O. Ozdikis, P. Senkul, H. Oguztuzun, Semantic expansion of hashtags for enhanced event detection in Twitter, in: Proceedings of the 1st International Workshop on Online Social Systems, 2012.
[28] A. Ahmed, Q. Ho, J. Eisenstein, E. Xing, A.J. Smola, C.H. Teo, Unified analysis of streaming news, in: Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 2011, pp. 267–276.
[29] H. Becker, M. Naaman, L. Gravano, Beyond trending topics: real-world event identification on Twitter, in: Proceeding of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM 11), AAAI Press, 2011, July.
[30] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, TX, USA, 1998, pp. 604–613.
[31] R. Weber, H. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: Proceedings of the 24th International Conference on very Large Data Bases, 1998, pp. 194–205.
[32] S. Petrovic, Real-Time Event Detection in Massive Streams, PhD Dissertation, Institute for Language, Cognition and Computation, School of Informatics, University of Edinburgh, UK, 2012.
[33] P. Indyk, R. Motwani, P. Raghavan, S. Vempala, Locality-preserving hashing in multidimensional spaces, in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, TX, USA, 1997, pp. 618–625.
[34] A. Dasgupta, R. Kumar, T. Sarlos, Fast locality-sensitive hashing, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 2011, pp. 1073–1081.
[35] M. Slaney, M. Casey, Locality-sensitive hashing for finding nearest neighbors, IEEE Signal Process. Mag. 25 (March) (2008) 128–131.
[36] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the 20th Annual Symposium on Computational Geometry, Brooklyn, NY, USA, 2004, pp. 253–262.
[37] B. Sharifi, M. Hutton, J.K. Kalita, Experiments in microblog summarization, in: Proceedings of the 2010 IEEE Second International Conference on Social Computing, 2010, pp. 49–56.
[38] S. Yardi, D. Boyd, Tweeting from the Town Square: Measuring Geographic Local Networks, 2010.
[39] M. Naaman, H. Becker, L. Gravano, Hip and trendy: characterizing emerging trends on Twitter, J. Am. Soc. Inf. Sci. Technol. 62 (May) (2011) 902–918.
[40] H. Kwak, C. Lee, H. Park, S. Moon, What is Twitter, a social network or a news media? in: Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 2010, pp. 591–600.
[41] I.P. Cvijikj, F. Michahelles, Monitoring trends on Facebook, in: Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011, pp. 895–902.
[42] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, Inc., New York, NY, USA, 1986.
[43] M.A. Russell, Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites, O'Reilly Media, Inc., 2011.
[44] A. Broder, On the resemblance and containment of documents, in: Proceedings of the Compression and Complexity of Sequences 1997, 1997, p. 21.
[45] H. Li, LSH: Theory and Application. JDL Seminar on Large-Scale Indexing and Search, 2010, July, Available at: http://www.haoli.me/uploads/9/0/6/7/9067538/jdl-lsh_seminar_lh.pdf
[46] M.S. Charikar, Similarity estimation techniques from rounding algorithms, in: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, Montreal, Quebec, Canada, 2002, pp. 380–388.
[47] D. Ravichandran, P. Pantel, E. Hovy, Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Ann Arbor, MI, 2005, pp. 622–629.
[48] M.T. Goodrich, R. Tamassia, M.H. Goldwasser, Data Structures and Algorithms in Python, Wiley Publishing, 2013.
[49] E. Fredkin, "Trie Memory" Communication of the ACM, September 1960, pp. 490–499.
[50] K.Y. Kamath, J. Caverlee, Content-based crowd retrieval on the real-time web, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, 2012, pp. 195–204.
[51] S. Goorha, L. Ungar, Discovery of significant emerging trends, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 2010, pp. 57–64.
[52] L. Uden, L.S.L. Wang, J.M.C. Rodrguez, H. Yang, I. Ting, The 8th International Conference on Knowledge Management in Organizations: Social and Big Data Computing for Knowledge Management, Springer Publishing Company, Incorporated, 2013.

**Shakirabanu Kaleel** is an IT professional having over 10 years of experience and currently serves as a graduate assistant and pursuing her Masters in Computer Science at Ryerson University, Toronto, Canada. She received her B.E. in Computer Science from Bharathidasan University, India. Her current active research is in the field of data mining in social networks.

**Abdolreza Abhari** is a professor in the Department of Computer Science at Ryerson University in Toronto, Canada where he has worked for past 10 years. He obtained his Ph.D. degree in computer science from Carleton University, Ottawa, Canada in 2003. He received his M.Sc. degree in software engineering in 1992 and B.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran in 1989. He worked as computer system consultant for industrial business segments from 1992 to 1998 in Iran, Netherlands, Germany and Canada. Dr. Abhari's research interests are data mining, web information retrieval, database and distributed systems, modeling and simulation and software engineering. He is a member of ACM and ACM SIGSIM.