Twitter Event Detection Under Spatio-Temporal Constraints

Gaolei Fei(✉), Yong Cheng, Yang Liu, Zhuo Liu, and Guangmin Hu

University of Electronic Science and Technology of China, Chengdu 611731, China
fgl@uestc.edu.cn, chengyong@std.uestc.edu.cn

Abstract. Billions of data spread on Twitter every day, which carries a lot of information. It is meaningful to mine the useful information and make it valuable. The purpose of Twitter event detection is to detect what happened in our real life from these unstructured data. We introduce the spatio-temporal information of tweets into event detection. The event detection can be divided into three steps in this paper. First, we use the space difference between event words and noise words and introduce the relationship between words, then we can build a model to separate event words and noise words. Then we define the similarity between event tweets from three different aspects, which make up for the shortcomings of existing methods. Finally, we construct a graph based on the similarity between tweets, and the graph can be divided into different event clusters to complete the event detection. Our method has achieved good results and can be applied to event detection in actual life.

Keywords: Twitter event detection · Noisy words identification · Spatio-temporal constraints · Condtional probability

1 Introduction

The number of active users in Twitter reaches 400 million, hundreds of million tweets are sent every day. These tweets record the details of events at the first moment. In these unstructured data, the goal of event detection is to find out tweets which describing events and extract the information we need from these event tweets, such as location, time and the key word of the event.

In the Twitter event detection, many researchers have proposed various methods. The general idea of these methods is to cluster the keywords or the text of tweets so that each tweet cluster corresponds to an event. For example, Doulamis et al. [1] uses tweet's sending time and the influence of Twitter users to define the similarity between words in tweets, and implements event detection by dividing words into different events. Dong et al. [2] combines the time and space information of tweets to get the similarity between tweets to complete event detection. This paper follows their idea that noisy words should obey the homogeneous Poisson process, and besides this, we establish a word network to extract noisy words more accurately. Ifrim et al. [3] uses the length and structural characteristics of tweets to cluster tweets. Caverlee et al. [4] regards the spatio-temporal information about tweets as signal, analyzes the characteristics of these noise signals, and applies many noise filters to remove noisy tweets, which can improve the quality of event detection.

However, the above methods base on text similarity and clustering can not applied to actual social media event detection. There are some problems in these methods.

First, in actual, more than 90% of tweets do not contain event information. many tweets are used to record the user's own life, express their own emotions and so on. These "noisy tweets" will affect the event detection results if we do not filter out them.

Second, there are a lot of tweets that do not contain event information, but their text may be similar to the tweets that describe the event, and these tweets may be clustered together with event tweets, which lead to a large number of "noisy tweets" in the clustering result.

Third, there also have some tweets describe the same event, but they do not contain common words and are considered completely different in their text. For example, $Tweet_1$ = "please do what you can to help the victims of the campfire in Paradise", $Tweet_2$ = "It breaks my heart to hear about people and animals losing their lives due to the California wildfires", although they have no common words, they all describe California fire events. However, existing methods usually can not cluster these tweets together, which may cause the mission of event information. How to solve this problem? We know that there may exist some tweets that describe the same event and have common words with $Tweet_1$ and $Tweet_2$, such as $Tweet_3$ = "Paradise, CA #wildfire #campfire @Paradise, California". We can build a model to measure the similarity between $Tweet_1$ and $Tweet_2$ through the intermediate tweet ($Tweet_3$), the problem can be solved in this way.

Aiming to solve these problems, we have proposed some methods. First, we study the difference between noisy words and event words, and find that noisy words are independent to each other and appear randomly in space. Using these features, we can separate noise words and event words to achieve the purpose of identifying noise tweets. Second, we use the spatiotemporal information of tweets as an important constraint on the measurement of tweets similarity. In this way, we can measure the similarity between tweets more comprehensively and accurately, and solve the second problem. Finally, to measure the similarity of tweets that are different in their text but belong to the same event, we introduce the co-occurrence similarity, using the co-occurrence of different words to measure the similarity of different tweets.

2 Problem Formulation

The input of Twitter event detection is tweet stream $T = \{T_1, T_2, ..., T_n\}$, where T_i denotes one tweet. The purpose of event detection is to divide T into different clusters so that each tweet cluster can correspond to an event in actual life. The idea of our method can be divided into three steps. First, filtering out "noisy tweets" in T and remain "event tweets". Secondly, We define the similarity between "event tweets" from three different perspectives. Finally, we construct a tweet similarity graph $G = (V, E)$, where graph vertex V denotes tweet, graph edge E denotes the similarity between tweets. By dividing this graph, the "event tweet" can be divided into different clusters, and each cluster can correspond to an event.

3 Noisy Words Identification

The first step in event detection is to remove noisy tweets. If all the words in T_i are noise words, then T_i can be regarded as a noise tweet. Hence, we need to identify noisy words first.

Noise words have two completely different characteristics from event words. Firstly, the appearance of noise words in different tweets are independent of each other. In contrast, the occurrence of words describing the same event is interrelated. Secondly, noise words appear in different regions with the same probability, that is, noisy words appear randomly in space. However, event words concentrates on the place where the event occurs. Therefore, we deem that noise words follow the homogeneous Poisson process in space, while event words do not follow. In short, if a word follows the homogeneous Poisson process, then we think this word is a noise word, and vice versa.

To measure whether a word w_i follows the homogeneous Poisson process, we can use Ripley's K function [5] to quantify. The Ripley's K function is as follows:

$$\widehat{K}(s) = V(A) \sum_{i \neq j} N(d_{ij} < s)/n^2 \tag{1}$$

Where s denotes the distance threshold between tweets, which is a experience value and the setting of s will be elaborated in the experimental part. $V(A)$ denotes the size of area A where the tweet stream is located, d_{ij} represents the Eucidean distance between two tweets that both contain w_i, and n is the number of tweets containing w_i. If w_i follows to the homogeneous Poisson process, then the result calculated by Eq. (1) will be πs^2. Because the result is related to s and not easy to measure, so we use the standardized K-funcation: $\widehat{L}(s) = \sqrt{\widehat{K}(s)/\pi} - s$ to standardize, and $\widehat{L}(s, w_i)$ is approximately equal to 0 if w_i follows the homogeneous Poisson process approximately. Thus, the proximity of $\widehat{L}(s, w_i)$ to 0 can be employed for evaluating how similar w_i follows the homogeneous Poisson process. We can define a threshold l and a tolerance limit β. If $\widehat{L}(s, w_i) < l - \beta$, then w_i follows the homogeneous Poisson process approximately and can be regarded as a noisy word. If $\widehat{L}(s, w_i) > l + \beta$, then w_i can be regarded as an event word. Finally, we think we cannot judge wether wi is a noisy word or an event word if $l - \beta \leq \widehat{L}(s, w_i) \leq l + \beta$.

It is not enough to judge whether w_i is a noise word by simply calculating it's standardized Replay's K function value. In our experiment, we also find that the selection of s value has some influence on the event word, but has little effect on the noise word. In this case, some event words may be mistakenly judged as noise words.

We know that words describing the same event are related to each other. If an event word is misjudged as a noise word, this misjudgment can be saved by other words that related to it. For a word w_i, we can use the conditional probability $P(i,j) = P(w_i|w_j)$ as the correlation strength of w_j to w_i, where $P(i,j)$ means the occurrence probability of w_i when w_j appears. In this way, we can create a graph $G_w = (V,E)$ to show the relationship between words. In graph $G_w = (V,E)$, V is vertices collection and each vertex represents a word, E is the edges between words and each edge denotes the correlation strength $P(i,j)$ between two words. In this way, V_i represents word w_i, and we can use V_i to judge wether w_i is a noisy word. We set the initial value of each vertex V_i to $\widehat{L}(s,w_i)$, then we update $V_i = V_i + \sum_{j=1}^{k} p(i,j) * V_j$, where V_j represents w_j related to w_i. In this way, whether a word is a noise word is not only affected by the $\widehat{L}(s,w_i)$ value, but also by the word associated with it. The specific algorithm is as follows. If w_i is an event word, then words related to it are most

Algorithm 1. Word attribute division

```
1: input:
   $T = \{T_1, T_2, ...\}$:tweet stream
   $T_i = \{geo, words\}$:each tweet information
   $n$:number of words (usually set to 3w-5w)
   $l, \beta$:$l$ is $\widehat{L}(s)$ value threshold, $\beta$ is the fuzzy bound.
2: Take out the most frequently occurring $n$ words and calculate the conditional
   probability $P(i,j) = P(w_i|w_j)$.
3: Take each word as a vertex $n_i$, the weight between the vertices is $P(i,j)$.
   vertex initial value $V_i = \widehat{L}(n_i, s)$
4: If $V_i \geq l + \beta$, set $V_i = 1$, indicating $n_i$ is an event word.
   If $V_i \leq l - \beta$, set $V_i = -1$, indicating $n_i$ is a noisy word.
   Otherwise set $V_i = 0$, indicating $n_i$ is unable to judge.
5: Starting from one vertex $n_i$, find all the vertexes $\{N_k\}$ that connected to $n_i$, update
   $V_i = V_i + \sum p(i,k) * V_k$ , repeat this process until all $V_i$ value have been updated.
6: Find $V_i \in [-1, 1]$ and continue with process 5 until all $V_i \notin [-1, 1]$.
7: output:
   $V_i (0 \leq i \leq n)$. $V_i < -1$ means $n_i$ is a noise word, otherwise $n_i$ is an event word.
```

likely event words that belong to the same event. Even if the $\widehat{L}(s)$ value of w_i "drop in" the scope of the noise word, according to the 5th step of Algorithm 1, this misjudgment can be saved.

4 Tweet Similarity

In Sect. 3, we can identify noisy words and event words. If T_i is all consisted of noise words, we can confirm that T_i does not contain event information and delete it, so that the remaining tweets are all event tweets. In this section, we define the similarity between tweets by merging three similarities in different aspects, that is the text similarity, the word time signal similarity under the spatio-temporal constraints, and the co-occurrence similarity.

4.1 Text Similarity

When calculating the similarity of the tweet text, we use TF-IDF (Term Frequency-Inverse Document Frequency) to assign weights to each word first, then convert each tweet into a vector, and finally use the cosine similarity to calculate the text similarity between tweets.

Supposing that two tweets T_i, T_j have a TF-IDF weight vector $\mathbf{X}, \mathbf{Y}$, then the text similarity between them is

$$S_{text}(i,j) = \frac{\mathbf{X} \cdot \mathbf{Y}}{|\mathbf{X}| \cdot |\mathbf{Y}|} = \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}} \tag{2}$$

4.2 Spatio-Temporal Similarity

Another situation is that although two tweets are similar in their text, they may not belong to the same event. Just considering text similarity as tweet similarity is not enough in this case. Tweets belonging to different events tend to have large differences in time or space. Therefore, it is necessary to add spatio-temporal constraints to the measurement of similarity between tweets.

First, we need to construct tweet words' signal. For two tweets T_1 and T_2, supposing that they have a common word w_i. Respectively taking the two tweets as the regional center and the event scope d as the radius, then counting the frequency that w_i appears in all the tweets in the two regions in each time period. The length of the time period is the time resolution Δt. In this way, w_i gets two time signals series from two tweets. If the two tweets describe the same event, then the two signals of w_i should have some correlation. The coefficient r^2 is used to measure the similarity of the two signals, and the larger value of r^2 indicates the higher similarity between this two signals.

$$r^2 = \left(\frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \right)^2 \tag{3}$$

Secondly, we add the spatio-temporal constraints to the similarity measurement. Assuming the distance between the two tweets is Δd. We connect Δd and the time resolution Δt together by adjusting Δt with Δd. Specifically, if two tweets are far from each other, then they should not belong to the same event, and the similarity between them should be as low as possible. We can reduce the similarity between tweets by making Δt smaller. That is, the bigger Δd is, the smaller Δt will be. In the same condition, the bigger the time resolution Δt is, the higher of the similarity between tweets will be. As shown in Fig. 1, the statistics "fire" and "lose" appear in 64 h. If resolution $\Delta t = 1\,\text{h}$, then $r^2 = 0.345$, and if $\Delta t = 4\,\text{h}$, $r^2 = 0.693$. It can be seen that the time resolution Δt can directly affect the similarity between word signals. In the same condition, the similarity between word signals can be smaller by reducing the time resolution. Therefore, the time resolution Δt can be adjusted by calculating the distance between tweets, then we can achieve the purpose of adjusting the similarity.

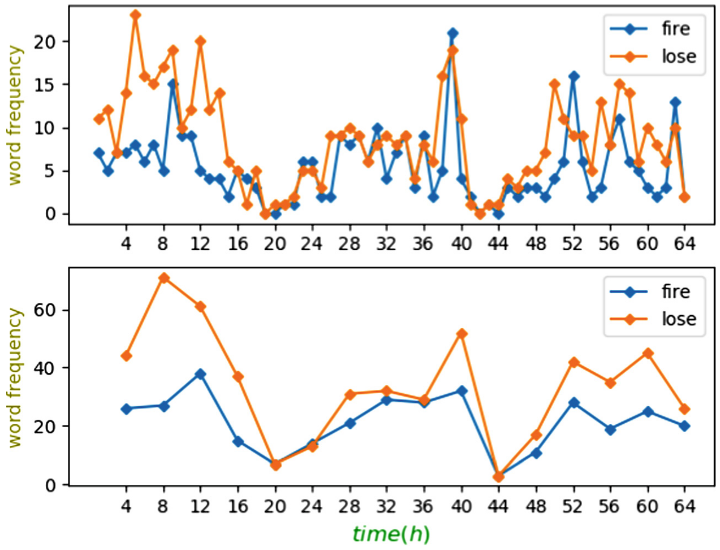

Fig. 1. Frequency of words at different time resolutions Δt

We define D_{max} as the maximum distance between two tweets in tweet stream, define T_{max} as the maximum time interval. According to the distance between two tweet Δd, then the time resolution Δt can be defined as

$$\Delta t = \frac{T_{max}}{6} / (log_{10}^{\frac{D_{max}}{100}} + 1 - log_{10}^{\frac{D_{max}}{\Delta d}}) \tag{4}$$

In Eq. (4), we set $\Delta d = 100\,\text{m}$ if $\Delta d < 100\,\text{m}$. If Δd is the minimal value, $\Delta t = \frac{T_{max}}{6}$, which means we can divide T_{max} into 6 segments at least. The word time signal contains at least 6 values, which guarantees the amount of basic information. If Δd becomes larger, then T_{max} is divided into more segments. For example, assuming $D_{max} = 100000$, $\Delta d = 10000$, $\Delta t = \frac{T_{max}}{24}$ will divide T_{max} into 24 segments. In this way, the time resolution Δt is adjusted by the space distance Δd.

After adding spatio-temporal constraints to the similarity measure, we also need to determine the value of event scope d. We are unable to get d without knowing the current specific event. We set $d = \{100, 1000, 10000, \cdots, D_{max}\}$, and traverse these values in turn to calculate similarity between word signals. We take the biggest value as the similarity of two word signals. If the two tweets describe the same event, then under this value, the similarity of the word signal is the highest. The reason is that if the size of the statistics area is larger than the event region, noise is added. If smaller, useful information is lost. So when the similarity is the highest, d is the value closest to the real event scope.

The last problem is that when two tweets have many common words, we take the highest similarity value as the similarity between the two tweets. The entire algorithm and implementation details are shown in Algorithm 2.

Algorithm 2. Calculating tweet similarity under spatio-temporal constraints

```
1: input:
   $T = \{T_1, T_2, ...\}$:tweet stream
   $T_i = \{timestamp, geo\}$:time and location information for
   each tweet
1: Calculate $D_{max}$ and $T_{max}$.
2: Find out $commonWords = \{w_i, i = 1, 2, \cdots, w_n\}$ for every two tweets in $T$,
   calculate their distance $\Delta d$, then calculate $\Delta t$ by equation (4).
3: For each common word $w_i$, respectively taking the two tweets as the regional center
   and the different event scope $d = \{100, 1000, \cdots, D_{max}\}$ as the radius, counting the
   frequency that $w_i$ appears on the time interval $\Delta t$, then we can get word signals in
   different scopes. Take the value with the highest similarity as the time signal
   similarity of $w_i$.
4: Perform the operation shown in step 3 for all common words in turn, taking the
   maximum value of the similarity as the similarity between the two tweets $S_{wordSignal}$.
5: output:
   Similarity between two pairs of tweets $S_{wordSignal}$
```

4.3 Co-occurrence Similarity

The main disadvantage of above method is that the similarity is always 0 if two tweets do not have a common word. That is to say, the calculation of similarity by text similarity or spatio-temporal similarity will become invalid in this situation. In fact, two tweets that are different in text may belong to the same event. Aiming at solving this problem, we propose another similarity measurement method called cooccurrence similarity—using the co-occurrence of different words to measure the similarity of different tweets.

For all tweets, we can use conditional probability to represent the strength of the association between two words. Let T_i words list be $\{w_i, i = 1, 2, \cdots, m\}$, T_j words list be $\{w_j, j = 1, 2, \cdots, n\}$. For all tweets, we calculate the $P(w_j|w_i)$ and $P(w_i|w_j)$ respectively, where $P(w_j|w_i)$ represents the occurrence probability of

w_j when w_i appears, $P(w_i|w_j)$ represents the occurrence probability of w_i when w_j appears. The strength of association between w_i and w_j is the maximum value of $P(w_j|w_i)$ and $P(w_i|w_j)$. With the strength of association between words, we can define the similarity between T_i and T_j which have no common words as S_{prob}

$$S_{prob} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} max(P(w_j|w_i), P(w_i|w_j)) \quad (5)$$

4.4 Comprehensive Measure of Tweet Similarity

In the above subsection, we have defined the tweet similarity measurements model which is applicable to every condition from three parts—text similarity, spatio-temporal similarity and co-occurrence similarity. Word signal similarity is a supplement to text similarity, and mainly used to this situation that two tweets are similar in their text but not belong to the same event. Therefore, these two similarity measurements need to be combined, we use the word signal similarity $S_{wordSignal}$. as the weight coefficient of the text similarity S_{text}.

Co-occurrence similarity is another supplement. It applies in this condition which the tweet text have no common words but they belong to the same event. We take the maximum value of $S_{text} * S_{wordSignal}$ and co-occurrence similarity S_prob as the similarity between two tweets.

$$S = max(S_{text} * S_{wordSignal}, S_{prob}) \quad (6)$$

5 Tweet Cluster Partition

In the above section, we have been able to calculate the similarity between two tweets and complete the second step of event detection. After defining the similarity between tweets, we can create a tweet similarity graph $G = (V, E)$, where V denotes tweets, E denotes the similarity between tweets. Using the Louvain algorithm to divide G, we can cluster the tweets that describe the same event.

Louvain is a community detection algorithm and it is very efficient. The time complexity of Louvain is $O(kN + E)$, where N is the number of vertices, E is the number of edges We use Louvain algorithm to divide G into multiple clusters. and each cluster is a description of an event. However, not all tweet clusters represent an event we need, we also need to filter out the tweet cluster that do not contain event information. First, if a cluster contain too few tweets (less than 3) should be deleted, because this is more likely to describe some small things or noise tweets, not the object of interest. Second, most of the tweets in the tweet cluster are sent from the same person should be deleted. In this case, it is likely to be an advertisement.

6 Simulation Results

6.1 Data Collection and Preprocessing

In order to evaluate the performance of our method, we use Twitter stream API to collect a total of $284k$ tweets in three days from 2018-11-17 to 2018-11-20 in California, USA. Each tweet is formed as $T_i = \{user\_id, text, user\_mentioned, hashtag, timestamp, geo, words\}$, where *user_mentioned* and *hashtag* are extracted from the tweet text, *geo* is the latitude and longitude information of the tweet sender, *timestamp* is the timestamp when the tweet is sent. *words* is obtained by tweet text segmentation, lemmatization, and filtering out stop words. We also need to remove some tweets that do not contain valid information. The rules are as follows

- The number of words in *words* is less than 2, which means the available information is too small.
- The words in *words* are all *user_mentioned* words, then the tweet does not describe the content of the event.

6.2 Filtering Out Noisy Words

we take the $n(n = 30000)$ words with the highest frequency to analysis. For each word w_i, we need to take out all the geographic coordinates that w_i appears and calculate their distance in pairs. For two points $A(LatA, LonA)$ and $B(LatB, LonB)$ on the earth, the distance between them is

$$d = 2R \arcsin(\sqrt{\sin^2(\alpha) + \cos(LatA)\cos(LatB)\sin^2(\beta)}) \tag{7}$$

where $\alpha = (LatA - LatB)/2$, $\beta = (LonA - LonB)/2$, $R = 6371\,\text{km}$. For all tweets containing the word w_i, Eq. 7 can be used to calculate the distance d_{ij} between two tweets.

The $\widehat{L}(s)$ value of the noise word is hardly affected by changing s, but the event word will be affected. The reason is that the distribution of event words is concentrated in the event occurrence area. The scope of event ranges from a few hundred meters to several tens of kilometers. Therefore, we select a list of s values, calculate $\widehat{L}(s)$ from 1 km to 36 km, and take the average value as the $\widehat{L}(s)$ value of the word w_i. Take California as an example, in Eq. (1), $V(A)$ denotes the area of California, $V(A) = 411000\,\text{km}^2 \text{s} = \{1\,\text{km}, 2\,\text{km}, \cdots, 36\,\text{km}\}$. For the noise words "love", "night" and the event words "wildfire", "death", the relationship between $\widehat{L}(s)$ and s is shown in the Fig. 2. There was a big wildfire in California on November 17, so "wildfire", "fire" are event words, and their $\widehat{L}(s)$ values are all above 0.8. The value of $\widehat{L}(s)$ fluctuated with the change of s value. Conversely, "night", "love" are "noisy words", their $\widehat{L}(s)$ value is close to zero, and their $\widehat{L}(s)$ values are almost unaffected by s value. In the algorithm I, we set $l = 0.6, \beta = 0.15$, we can initially judge that "night" and "love" are noise words, "wildfire" and "fire" are event words. In the actual situation, there will be some words' $\widehat{L}(s)$ value in the interval $[l - \beta, l + \beta]$, and it is impossible

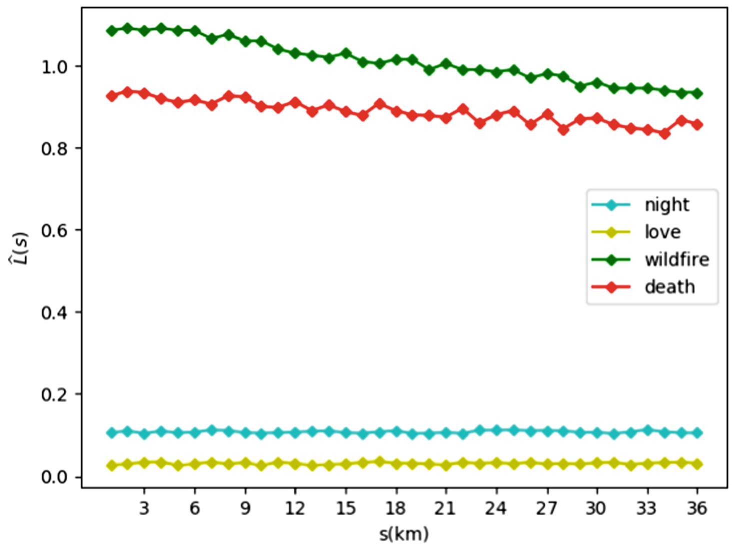

Fig. 2. $\widehat{L}(s)$ value for different s values

to judge whether the word is a noise word or cause misjudgment. We take the average of $\widehat{L}_s$ as the initial value of the word w_i, and bring it in the algorithm I to calculate the final value V_i of the word w_i. After testing, algorithm I can effectively save this misjudgment under normal circumstances.

The experimental results show that among the 30,000 words with the highest frequency, only 1587 words are event words, and words over 94% are noise words. After removing the tweet without any event words, the number of tweets is reduced from 284k to 36k.

6.3 Tweet Similarity

First, we calculate the tweet text similarity S_{text}, we use the TF-IDF method in *sklearn* to calculate the word weight, then bring the result into the Eq. (2) and calculate the text similarity between two tweets.

Secondly, we measure the similarity $S_{wordSignal}$ between tweets by constructing word signal sequence. The distance of the longest distance in the tweet stream is $D_{max} = 989.9\,\text{km}$, and the longest time gap is $T_{max} = 72\,\text{h}$. For T_i, T_j, suppose they have two common words w_i, w_j, the distance between them is $\Delta d = 1000\,\text{m}$, then the time resolution $\Delta t = 6\,\text{h}$ according to the equation (4). For one word w_i, dividing $T_{max} = 72\,\text{h}$ into 12 segments, the event scope $d = \{0.1\,\text{km}, 1\,\text{km}, 10\,\text{km}, 100\,\text{km}, D_{max}\}$. Respectively taking the two tweets as the regional center and the event scope d_i as the radius, then counting the frequency that w_i appears in each time period Δt, finally we get time signal series of w_i. The similarity of these time series signals are calculated by the Eq. (3), and the maximum value of similarity under different d_i is taken as the similarity value. Then we calculate the similarity of the word w_j, and take the maximum value as the similarity of T_i and T_j.

Thirdly, we use conditional probability to calculate tweet similarity. The key point is to find out the probability of w_j when the word w_i appears. There are two ways to achieve this, one is to complete the statistic by traversing each tweet, and the speed is slow. The second is to use the idea of FP-growth to build a tree structure, which is fast. Finally, the similarity between the two tweets is determined by Eq. (6).

6.4 Tweet Partition and Event Extraction

The previous section achieve a measurement of similarity between two pairs of tweets. After denoising, tweets stream remain only $36k$ tweets. With each tweet as a vertex, the similarity between the tweets as the edge, and a tweet similarity graph is constructed. We delete edges with a similarity less than 0.05, which not only prevents the Louvain algorithm from combining the low-similar tweets, but also effectively reduces the amount of computation. In this way we construct

Table 1. California Top 3 event (with denoising + three similarity measures)

event1	Time	2018-11-17 03:59
	Location	[−120.10, 35.14]
	Key words	fire, lose, paradise, california, wildfire, smoke, heart, forest, death, burn, campfire, angeles
	Key tweets	1. Pray for the citizens of California, Fires to the east and south 2. #SanFrancisco #california #airquality #campfire @ San Francisco, California 3. It breaks my heart to hear about people losing their lives due to the wildfires
event2	Time	2018-11-18 00:05
	Location	[−118.12, 34.00]
	Key words	celebrate, birthday, november, day, mickey, mouse, 90th, happen, california, love, anniversary, great
	Key tweets	1. Mickey Mouse turns 90 today! Happy Birthday Mickey! 2. Happy birthday Mickey! #mickey90 #happybirthdaymickey #mickeymouse #disney 3. Happy 90th Birthday Mickey Mouse! And Happy Birthday Minnie Mouse!
event3	Time	2018-11-18 06:20
	Location	[121.81, 39.73]
	Key words	trump, california, californiafires, impact, woolseyfire, presidential, visit, areas, diss, forest, management, again
	Key tweets	1. PRESIDENTIAL VISIT: @realdonaldtrump toured areas impacted by the #CampFire 2. Trump in California and he dissed forest management again lmao 3. #makeamericarakeagain #californiafires #rake #trump @ Paradise, California

Table 2. California Top 3 event (without denoising+use text similarity only)

event1	Time	2018-11-17 04:41
	Location	[−119.56, 35.31]
	Key words	fire, lose, paradise, california, day, forget, return, ag, smoke, heart, camp, air
	Key tweets	1. Our hearts go out to those working to recover from the Woolsey Fire 2. This is what California calls...A Beautiful Disaster 3. A few shots from the former town of Paradise, wiped from existence by fire last week
event2	Time	2018-11-18 07:38
	Location	[−118.26,34.51]
	Key words	celebrate, birthday, november, day, love, time, friend, mickey, night, 90th, happen, anniversary
	Key tweets	1. Happy 90th Birthday Mickey! I'm so happy we could celebrate with you today in disneyland 2. Had such a wonderful day with friends celebrating Mickey 3. Celebrated Mickeys 90th Birthday at Walts Barn!!
event3	Time	2018/11/17 3:59
	Location	[−118.28, 34.15]
	Key words	car, fire, right, lane, traffic, fairway, stop, delay, lose, ave, destroy, center
	Key tweets	1. Car fire on the right shoulder in #Lynwood on 105 EB at Long Beach Blvd 2. !! sigalert !! the two right lanes are closed because of a car fire 3. Vehicle on fire in #Salida on Hwy 99 NB before Hammett Rd

a sparse graph of $36k$ nodes and $356k$ edges. Using the Louvain algorithm to divide tweets into several tweet clusters, using the tweet cluster filtering and event information extraction methods in Sect. 5, we can get the event detection results as shown in Table 1. For comparison, Table 2 shows the results obtained by not using tweets denoising and using only text similarity as a measure. Take the largest top K tweet clusters as important events. Here we take the first three clusters, extract the event information, and compare it with the real event. The real events are

– On 2018.11.17, two vast wildfires ravaged parts of California, killing at least 66 people.
– 2018.11.18, the 90th birthday of Disney Mickey Mouse.
– US President Trump arrived in California on the afternoon of Saturday (17th) to learn about the serious damage caused by wildfires.

From the results in the table, we can see that we can extract keywords more accurately if we filter out noisy tweets and use three methods to measure tweets similarity, and all three things are successfully detected. In Table 2, the event detection result key words is mixed with a large number of words unrelated to the current event because of lacking denoising. At the same time, because the lack of comprehensive measurement of the similarity between tweets, event3 puts tweets that describe the traffic accidents together. In fact, these traffic accidents are not the same event.

7 Conclusion

This article focus on Twitter event detection, and put forward our own ideas in filtering out noise tweets and defining the similarity between tweets. In order to remove noise words more accurately, we not only quote the $Replay'sK$ function to measure the spatial distribution of noise words, but also establish a word graph to comprehensively judge word attributes by their related words, which greatly reduces the probability of word misjudgment. In order to define the similarity of tweets, we quantify the influence of the spatiotemporal information of tweets, and propose how to measure the similarity between tweets when they have no common words. These ideas have also achieved good results with less noises and higher accuracy in practice.

References

1. Doulamis, N.D., Doulamis, A.D., Kokkinos, P., et al.: Event detection in Twitter microblogging. IEEE Trans. Cybern. **46**, 1–15 (2015)
2. Dong, X., Mavroeidis, D., Calabrese, F., et al.: Multiscale event detection in social media. Data Min. Knowl. Discov. **29**(5), 1374–1405 (2015)
3. Ifrim, G., Shi, B., Brigadir, I.: Event detection in Twitter using aggressive filtering and hierarchical tweet clustering (2014)
4. Liang, Y., Caverlee, J., Cao, C.: A noise-filtering approach for spatio-temporal event detection in social media. In: Hanbury, A., Kazai, G., Rauber, A., Fuhr, N. (eds.) ECIR 2015. LNCS, vol. 9022, pp. 233–244. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16354-3_25
5. Dixon, D.P.M.: Ripley's K Function. Encyclopedia of Environmetrics (2006)
6. Patil, M., Chavan, H.K.: Event based sentiment analysis of Twitter data. In: IEEE Conference Record #42656; IEEE Xplore ISBN 978-1-5386-3452-3
7. Sato, K., Wang, J., Cheng, Z.: Credibility evaluation of Twitter-based event detection by a mixing analysis of heterogeneous data. IEEE Access **7**, 1095–1106 (2018). https://doi.org/10.1109/ACCESS.2018.2886312
8. Shi, L.-L., Liu, L., Wu, Y., Jiang, L., Hardy, J.: Event detection and user interest discovering in social media data streams. IEEE Access **5**, 20953–20964 (2017)
9. Shi, L., Wu, Y., Liu, L., Sun, X., Jiang, L.: Event detection and identification of influential spreaders in social media data streams. Big Data Min. Anal. **1**(1), 34–46 (2018). ISSN 2096-0654 03/06
10. Kala, T.: Event detection from text data. Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague, May 2017