Lifecycle-Based Event Detection from Microblogs

Lin Mu
University of Science and Technology of China
P.O. Box 230027
China
mulin@mail.ustc.edu.cn

Peiquan Jin*
University of Science and Technology of China
P.O. Box 230027
China
jpq@ ustc.edu.cn

Lizhou Zheng
University of Science and Technology of China
P.O. Box 230027
China
zhenglz@mail.ustc.edu.cn

En-Hong Chen
Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China
P.O. Box 230027
China
cheneh@ustc.edu.cn

Lihua Yue
Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences
P.O. Box 230027
China
llyue@ustc.edu.cn

ABSTRACT

Microblog like Twitter and Sina Weibo has been an important information source for event detection and monitoring. In many decision-making scenarios, it is not enough to only provide a structural tuple for an event, e.g., a 5W1H record like <who, where, when, what, whom, how>. However, in addition to event structural tuples, people need to know the evolution lifecycle of an event. The lifecycle description of an event is more helpful for decision making because people can focus on the progress and trend of events. In this paper, we propose a novel method for efficiently detecting and tracking event evolution on microblogging platforms. The major features of our study are: (1) It provides a novel event-type-driven method to extract event tuples, which forms the foundation for event evolution analysis. (2) It describes the lifecycle of an event by a staged model, and provides effective algorithms for detecting the stages of an event. (3) It offers emotional analysis over the stages of an event, through which people are able to know the public emotional tendency over a specific event at different time periods. We build a prototype system and present its architecture and implemental details in the paper. In addition, we conduct experiments on real microblog datasets and the results in terms of precision, recall, and F-measure suggest the effectiveness and efficiency of our proposal.

CCS CONCEPTS

• **Information system → Information retrieval → Retrieval tasks and goals** → Information extraction

KEYWORDS: Event evolution, Event detection, Microblog, Lifecycle

ACM Reference format: L. Mu, P. Jin, L. Zheng, E. Chen, and L. Yue. 2018. Lifecycle-Based Event Detection from Microblogs. In *The 2018 Web Conference Proceedings (WWW 2018), April 23-27, 2018, Lyon, France*, ACM, New York, NY, 8 pages. DOI: https://doi.org/10.1145/3184558.3186338

ACM ISBN 978-1-4503-5639-8/18/04.
DOI: https://doi.org/10.1145/3184558.3186338

1 INTRODUCTION

Microblog platforms have been one of the major sources for new events detection and spreading. For example, Sina Weibo as the most popular microblogging platform in China involves over 280 million users, and over 1,000 tweets are posted every second. Motivated by the massive fresh information generated by microblog users, many works on event detection and analysis on microblogs have been conducted in recent years [1, 2].

However, previous studies mainly focused on extracting structural tuples of events, e.g., extracting the 5W1H (who, where, when, what, whom, how) information [1]. In addition to event structural tuples extraction, some studies paid attention to the evolution analysis of events. In the literatures [3-5], researchers performed evolutionary analysis for events by sorting events according to the timeline, or using other simple rules. These studies cannot grasp the development stage of events. On the other hand, an event usually has a developing stage in the real world, i.e., from birth to death, which is similar to the lifecycle of people. The lifecycle information of an event is very useful in information mining and decision making. For example, company managers can make specific decisions according to the developing stage of the events related to products. Basically, the lifecycle of an event can be defined as a five-stage process including a budding stage, a developing stage, a peak stage, a recession stage, and a pacification stage, as shown in Figure 1.

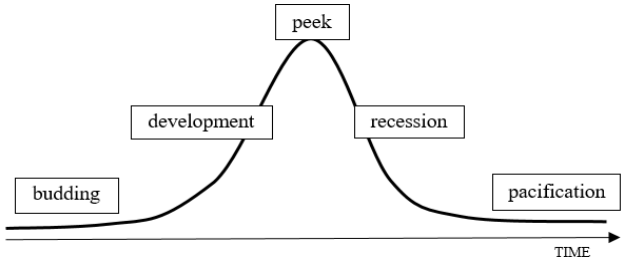

Figure 1. Lifecycle of an event

Event evolution has been studied in some previous works [6, 7]. For example, in [7] the researchers defined three processes, namely emerging, growing, and fading, to describe event evolution. However, they only described the changes of event hotness. In addition, the three processes defined in [7] cannot indicate the exact stage in the lifecycle of the event. On the other hand, they

used vectors to represent events, which were not able to the semantic details of events, e.g., the 5W1H details of events [1]. On the contrary, we propose the algorithm in this paper is not only able to describe the developing stages of event, but also able to provide detailed 5H1H information for each event.

Event evolution stage extraction aims to extract the developing stages of events. However, current algorithms in event extraction only focus on extracting a summary of events but not on their evolutional processes. The challenge lies in the linking and partitioning stage of extracted events. For instance, if there is an event tuple extracted from a collection of microblogs, we should first link the event tuple to the same event that occurred before, and then determine which stage of the lifecycle this event belongs to. The current event evolution extraction works have the following insufficient. First of all, they were not able to present the stage evolutional processes and emotion for events. In addition, they did not consider event types when extracting events from microblogs and did not provide a fine-grained description for the extracted events.

In this paper, we propose a novel method for detecting and analyzing the evolution process and emotional evolution of events on microblogging platforms. The major contributions in this paper are as follows:

(1) We propose to incorporate event type into the event tuple extraction. Inspired by the studies in the news-report area that describe an event based on the news features [1], i.e., when, where, who, whom, what, and how, we consider to detect the news features of events from microblogs.

(2) In order to grasp the development of events from the macro, we describe the evolution of events based on a five-stage lifecycle model that consists of five stages: budding, development, peek, recession, and pacification. We consider detecting the lifecycle stage of events from microblogs.

(3) The public emotional tendency to an event varies with time. Based on the extracted stages of an event, we develop a visual interface to monitor the public emotional evolution for each stage of specific events.

(4) We develop a prototype system to demonstrate the feasibility of our proposal. In addition, we conduct experiments on real microblog datasets. The results in terms of precision, recall, and F-measure suggest the effectiveness and efficiency of our proposal.

2 RELATED WORK

In this section, we summarize existing works. There are mainly three research areas that are closely related to our work, i.e., event extraction on microblogs, event evolution extraction, and event sentiment extraction.

2.1 Event Extraction on Microblogs

Event extraction on microblogs is a popular research focus in the field of information extraction. Most previous works in this field focused on detecting certain types of events [8] or events in open domains [9-10], and many approaches have been proposed such as LDA-based topic modeling [13], text classification and clustering.

The standard LDA model proposed by Blei et al. [13] was first used to extract the topic of the text. In [16] the author used a variety of methods to train the standard topic model in the microblogging text, and compared to the effectiveness and efficiency of these methods. Ritter et al. [9] extracted significant events in open domains based on topic models, combining with a named entity tagger and sequence labeling techniques. In [18], the authors used the improved method based on the LDA theme model and the recurrent Chinese Restaurant Process to extract topics and events. Other topic model based approaches were proposed in [19-20].

Text clustering is another popular approach for extracting events on microblogs [11-15]. It first extracts features like single words, hashtags [12], n-grams, or burst n-grams, and then inputs the extracted features into a similarity-based clustering algorithm extract events. In [15] the author proposed a method called EDCOW (Event Detection with Clustering of Wavelet-based Signals), which first extracts the event burst based on the wavelet theory and then clustered to form events with the graph partitioning technique clustering. A system called STED [21] also used the graph to divide the clustering method to extract events. The system first extracts the event-related keyword phrases, and then used the clustering method to get some smaller tweets (tweet mini-clusters). In the paper [11], the author first extracts the keywords that can express the event in the fixed time interval, and then use the bottom-up hierarchical clustering method to cluster these key words. There are also many algorithms to use deep learning to extract events [22, 23].

In this paper, we also employ the clustering technique for extracting events. Our approach is based on the agglomerative hierarchical clustering, which has been used in [11] to extract events based on given keywords. Differing from [11], we emphasize the importance of event type (the distribution over named entities) to compute the similarity between microblog posts in each cluster. In addition, we do not use features like burst words as they failed to extract events mentioned in few microblog posts [15, 21].

2.2 Event-Evolution Detection

Research work on Topic Detection and Tracking (TDT) [28] has attracted great attention in recent years. There are many studies on the evolution of events. For example, [25] took the evolution relationships at the topic-level into consideration to discovery temporal patterns of popular events from text streams. In [27] adopted a vector space model and TD-IDF to generate the document vectors, and the similarity of documents was measured by cosine similarity to describe the evolution relationships between the events, supplemented by the features of temporal proximity and document distributional proximity. Inspired by machine learning and rules based model, [24] extracted events from the level of the sentences, determined the evolutionary relationship between two events and demonstrated a new model of information retrieval and summary generation. The similarity between the content of the documents was incorporated with other features to evaluate the strength of the evolutionary relationship between news events. One of the greatest features of the research in this direction is the common use of the vector space model, the TF-IDF formula and cosine similarity to measure the similarity of the content of the documents. In addition to the research [27] on the evolution relationships between the events within a news topic, Wei et al. [26] focused on how to divide the news stories into the related news events and proposed an improved TF-IDF model on the basis of original one that it could be more effective in the discovery of news events from the corpus. There is also some work that was the evolution of real-time extraction events. In [7], a subgraph-by-subgraph incremental tracking framework was proposed for event monitoring. A skeletal graph was designed to summarize the information within a fading time window in a dynamic network. Cai et al. proposed [4] used

four event operations (i.e., create, absorb, split and merge) and designed an event index structure of multi-layer inverted list to manage and discover evolving events over Twitter streams. In [6], the researchers defined three processes, namely emerging, growing, and fading, to describe event evolution.

However, the above work has the following problems. First they used vectors or some key words to represent events, which were not able to the semantic details of events, e.g., the 5W1H details of events [1]. Secondly, although they can extract the evolution of the event, they don't have a model can clearly give the event at what stage.

2.3 Event Sentiment Analysis

Opinion mining and emotional analysis has been a hot research topic in recent years. Basically, sentiment analysis consists of five types of tasks [29], i.e., document-level sentiment analysis, sentence-level sentiment analysis, aspect-level sentiment analysis, comparative sentiment analysis, and sentiment lexicon acquisition. Different approaches have been proposed to realize the goals of those tasks. The first approach is called semantic-based approach [30, 23, 36], which performs sentiment analysis on the basis of rules and sentiment lexicons, and the second approach is machine learning-based methods [34, 35], which regards sentiment analysis as a binary or multi-class classification task and uses common classification methods. Most of the previous works on sentiment analysis can be put in the above two types, but there are also some other works using different methods [31, 32].

In recent years, the deep learning algorithm has been applied to the field of natural language processing, access to the results better than the traditional model. There are also many researchers who apply deep learning to various emotional analysis tasks [37]. Most of the existing work was from a variety of granularity to extract the text of the emotional tendencies, however, microblogging events will continue to evolve, so the user's emotional tendencies will change with the evolution of the event, our aim is to extract the user's emotional evolution of the microblogging event.

3 FRAMEWORK OF LIFECYCLE-BASED EVENT DETECTION

In this section we describe the features of the event evolution phase extraction algorithm. Figure 2 shows the process of our method. The key modules include event tuple extraction, event tuple linking, event lifecycle detection, and emotional evolution analysis.

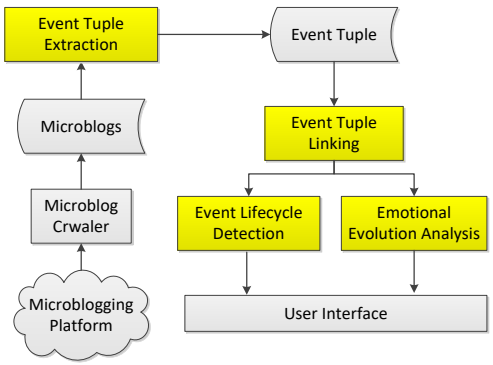

Figure 2. Framework of lifecycle-based event detection

3.1 Event Tuple Extraction

First of all, we pre-process the raw microblogging data set, including word segmentation and removing stop words. In order to facilitate real-time microblogging data processing, we divide the crawled microblogging data into slices. Each slice collects all the microblog post published in the same time interval (e.g., one hour or one day) [9]. These microblogging slices are processed in batch in a chronological order.

Next, we extract event tuples from sliced microblog data. The extraction of event tuples is based on event type, which is described as follows.

3.1.1 Event Type. The definition of event type is as follows:

Definition 1. ***Event Type.*** *Given a collection of microblog post T which is obtained by one event query word, the event type is defined as a* quadruple *<p_l, p_n, p_o, p_t>, in which p_l, p_n, p_o, p_t represent the importance of location, person name, organization and time entity in the collection respectively. Note that the sum of those four probabilistic values must be equal to one, i.e., $p_l+p_n+p_o+p_t$ =1.* ■

In order to make use of event type in extracting event tuples, we first extract event type, i.e., the probabilistic quadruple < p_l, p_n, p_o, p_t >. Given a microblog post collection, we represent it as a feature vector x and then employ the Multinomial Logistic Regression method to train the model, as shown in (1). The result $p_i = p(y = i|x^{(i)}, w)$ where i = l, p, o and t for different named entity categories is used as the probabilistic distribution [1].

$$p_i = p(y = i|x^{(i)}, w) = \frac{e^{w_k{}^T * x^{(i)}}}{\sum_k e^{w_k{}^T * x^{(i)}}} \quad (1)$$

Finally, we get a quadruple < p_l, p_n, p_o, p_t>, which is outputted as the event type. We will further use this quadruple to perform event tuple extraction.

3.1.2 Event Tuple Clustering. The calculation of similarity is a crucial part of the clustering process. A good similarity calculation method will make the effect of clustering significantly improved. Because of the different types of events, the distribution of different types of named entities is not the same; therefore, we use the named entity information to enhance the effect of clustering.

After we have extracted the event type, which is represented as the probabilistic distribution over different categories of named entities, we use it to calculate the similarity between microblog posts, and further perform clustering. Our calculation of similarity consists of two parts, the normal cosine similarity between terms in the microblog posts and the similarity between named entities.

(1) *Normal Term Similarity*: The normal term similarity is based on a bag-of-word model that represents each microblog post as a vector of terms. We employ the basic cosine similarity between two term vectors as the normal term similarity, which is denoted as sim_t.

(2) *Named Entity Similarity*: Another important part of similarity between microblog posts is the named entity similarity. As different categories of named entities may play different roles in different types of named entities, we propose to adjust the weights for the categories of named entities, e.g., to increase the weight of location similarity for *location*-based events like earthquake, or to increase the weight of *organization* similarity for *organization*-based events like *enterprise bankruptcy*. The named entity similarity between two microblog posts is calculated by (2).

$$Sim_e^T(\mathrm{m},\mathrm{n}) = \sum_i \left[p_i * \sum_j \sum_k Sim\left(E_{i,m}^T(j), E_{i,n}^T(k)\right) \right] \quad (2)$$

Here, $Sim(E_{i,m}^T(j), E_{i,n}^T(k))$ is the similarity between two named entities. This similarity is calculated based on the Minimum Edit Distance (MED). $p_i(i = l, n, o, t)$ represents the weight of each named entities.

(3) Overall Similarity: The overall similarity in our clustering process is the weighted sum of the normal term similarity and the named entity similarity, as shown in (4). Here, β is a trade-off between the two similarities.

$$Sim = \beta * Sim_t + (1 - \beta) * Sim_e \quad (3)$$

Because it is not possible to know the number of events contained in a microblogging data set in advance, we employ an agglomerative hierarchical clustering method for event tuple extraction. Through microblogging clustering, we get a lot of events microblogging clusters, and the microblog posts in each cluster are associated with the same event.

3.1.3 Event Detail Extraction. Through the previous section, we get the collection of microblogging event clusters, i.e., event tuple set. Next, for each cluster, we extract the 5W1H (when [38], where [39], who, what, whom) information, and the complete 5W1H element is often helpful for accurately describing an event. The process of event detail extraction is shown in Fig. 3[1].

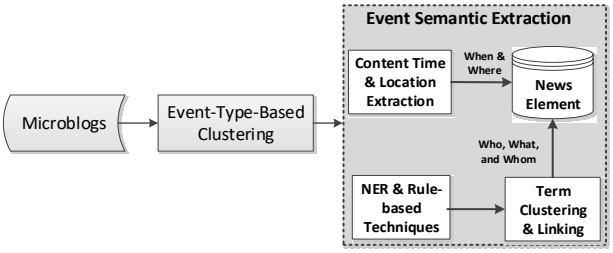

Figure 3. Process of event detail extraction

3.2 Event Tuple Linking

As an event will evolve over time, after we extract the event tuples of different microblogging slices, we link the event tuples that describe the same event. For instance, if we extract an event tuple on July 6 to describe an explosion event, this explosion event will continue to develop. On July 7, we also extract an event tuple describing the explosion. In this case, we need to link the event tuples describing the same event.

The detailed event tuple link algorithm is shown in **Algorithm 1**. Given the microblogging slice at time t_i, we first get the set of event tuples, represented by $\mathrm{eventTupleSet}_i$. For each event tuple in $\mathrm{eventTupleSet}_i$, we calculate the similarity of the event tuple to previous events, find the most similar event, and then link the event tuple to that event. If there is no similar event, we create a new event and add attach the event tuple to the newly created event.

When calculating the similarity of two events, we consider the following two kinds of similarity:

(1) Spatiotemporal Similarity. As the subject, time, and location information are specifically important for describing an event, we calculate the similarity based on *who*, *where*, and *when* elements in the extracted 5W1H information of event tuples. We also use the event type to adjust the weight of *who*, *where*, and *when* in similarity measurement.

(2) Semantic Similarity. The semantic information of an event is also crucial to distinguish it from other events. Thus, we extract the semantic information of an event by analyzing the textual words in event tuples, and then calculate the textual similarity among event tuples.

Algorithm 1. Event Tuple Link

Input: $eventTupleSet_i$: microblogging event tuple set at time t_i;
eventSet: microblogging event set
Output: *eventSet*: the revised microblogging event set
Preliminary: *threshold* is a pre-defined value for event similarity.

```
for each et in eventTupleSet_i do
    e ← the event in eventSet that is most similar to et;
    if similarity(e, et) > threshold then
        Add et into the tuple list of e;
    else
        Create a new event en;
        Add Add et into the tuple list of en;
        Add en into eventSet;
    end if
end for
return eventSet
```

3.3 Event Lifecycle Detection

Figure 4 shows the representation framework of an event. Each event has a unique ID and a set of event attributes. It also has a unique lifecycle that is five-bit structure indicating the current evolution process of the event. For example, <11000> means that the event is at the development stage. An event has a list of event tuples that are linked by the event tuple linking algorithm. All event tuples are arranged along the timeline and each tuple has an indicator describing what stage it belongs to.

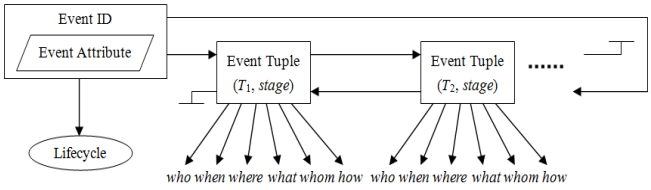

Figure 4. Event representation

The lifecycle information of events is of critical values in decision making. For example, when the event of Samsung smartphone explosion was reported on microblogging platforms, it developed very fast and brought negative impact to Samsung. On the other hand, if Samsung can know the evolution process (lifecycle) of the event in time, they can set out effective actions to lower the negative impact of the event.

The detailed event lifecycle detection algorithm is shown in **Algorithm 2**. Give a microblogging event tuple set $eventTupleSet_i$ and an event $event_{ij}$ at time t_i. First of all, we utilize the event tuple linking algorithm to get a number of event tuples *elist* that describe $event_{ij}$. Next, we extract the lifecycle $curStage_{ij}$ of $event_{ij}$ in the current time, then we assigned *elist* to $curStage_{ij}$, and update $event_{ij}$ with $curStage_{ij}$.

Algorithm 2. Event Lifecycle Detection

Input: $eventTupleSet_i$: microblogging event tuple set at time t_i;
$event_{ij}$: The jth microblogging event at time t_i
Output: $event_{ij}$ the jth revised microblogging event at time t_i

1. **for each** *et* in $eventTupleSet_i$ **do**
2. | **if** event link *et* and $event_{ij}$ success **then**
3. | | Add *et* to *elist*;
4. | **end if**
5. **end for**
6. $curStage_{ij}$ ←Detection $event_{ij}$ lifecycle with $event_{ij}$ update *elist*;
7. *elist* assigned to $curStage_{ij}$;
8. update $event_{ij}$ with $curStage_{ij}$;
9. **return** $event_{ij}$;

The key issue of event evolution detection is to determine the right stage of an event. Sometimes we need to predict the stage of an event in future. In our algorithm, we use the popularity of events to detect the event lifecycle. We define the popularity of an event in terms of the following features:

(1) The forwarding number and commenting number as well as the total number of related microblog posts are used to measure the popularity of an event.

(2) If users' emotional tendency towards an event changes dramatically, it implies that the evolutional stage of the event may change.

(3) People cannot always focus on one specific event. When a new and interesting event happens, it will attract user attention and change the evolutional stage of current events.

(4) When the locations that are embedded in the event tuples change, it usually implies the change of the evolutional stage of the event.

The above method is actually a rule-based approach. In our future work, we will consider other kinds of approaches such as machine learning approaches in outlier detection [17].

3.4 Emotional Evolution Analysis

The public emotional tendency over a certain event will change with time, rather than staying on the same state. Thus, it is helpful to track the emotional evolution of events in decision making. In this paper, we take three steps to extract the emotional evolution information of an event.

(1) First, We extract the microblogging event tuples from each microblogging slices and extract the emotional tendencies of the event tuple based on a given sentiment dictionary.

(2) Second, we map the event tuple to a developing stage of the event, as shown in the event representation in Fig. 4.

(3) Finally, we compute the overall emotional polarity for each stage of an event based on the emotional tendencies of the event tuples linked within the stage. We use a weighted sum to aggregate the emotional tendencies of the event tuples in a stage, in which we put high weights for recent event tuples.

4 SYSTEM IMPLEMENTATION

Based on the algorithm presented in Section 3, we developed a prototype system called EventSys. Figure 5 shows the software architecture of EventSys.

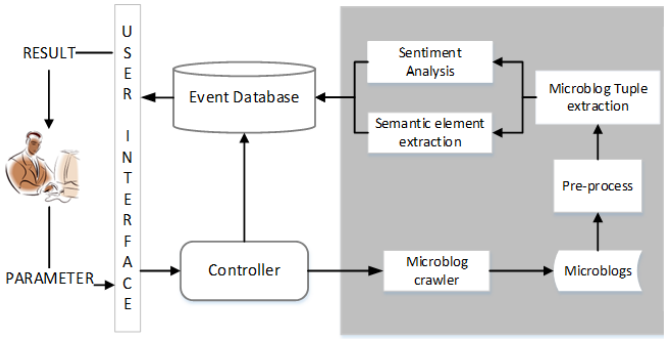

Figure 5. Structure of EventSys

The dataset crawled from the Sina Weibo by using 18 event keywords such as explosion, earthquake, election, and acquisition. There are 5 million microblog posts from Feb. 2013 to Mar. 2013.

After crawling the microblog posts, we first divide the data set into microblogging slices in days, and then preprocess the posts. The detailed steps are as follows:

(1) Remove the unnecessary tags such as *http* addresses and tags with the "@" symbol.

(2) Word segmentation and part of speech tagging for microblogging texts. Here we use the word segmentation tool NLPIR.

(3) Remove the posts that do not contain name entities.

Given an event keyword related microblogging set, we preprocess microblog posts. Then we extract the event type corresponding to the event in the microblogging set, assuming that the keyword is unclear. Finally, we cluster the microblogging set based on the event type, and get a number of event tuples related to an event word after clustering. The algorithm is described in detail in Section 3.1.

For semantic element extraction, we use the algorithms proposed in [1] to extract the 5W1H information for events. For the sentiment analysis module, we take three steps to extract the emotional evolution information of events. First, we use the TF-IDF method to calculate the similarity of two event tuples, and link the event tuples. Next, the emotional words in the event tuples of each event stage are extracted based on a given sentiment-word dictionary. Finally, we organize an event emotional appended a visualized emotional tendency.

Figure 6 shows the snapshot of the system. Zone A shows a list of all the events associated with the keyword. The event is sorted by the number of related microblogs. You can view the details of the event by clicking on the event. In Zone B, we can see that the events occurring mostly in the eastern coastal areas of China. The right part in Figure 6 shows the emotional evolution of the event, where different kinds of emotional information are presented, including the static statistical emotion, the dynamic emotional tendency, and the supported microblog posts.

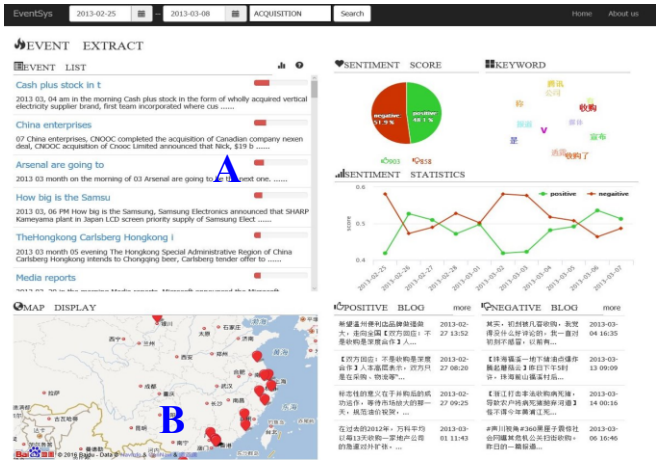

Figure 6. Snapshot of EventSys

5 PERFORMANCE EVALUATION

5.1 Dataset

In the experiment, we prepare two datasets by crawling posts from Sina Weibo (http://weibo.com). Given query words that are related to one or more events, we crawler microblogs that contain those query words. In our work, we consider two heterogeneous data sets: the first dataset consists of posts obtained by crawling posts which contain the given query words. Another dataset is composed of posts describing some specific events. We symbolize the two datasets as DS1 and DS2, as described below.

DS1. The first dataset DS1 contains collections of more than 450K posts crawled by event keywords from Feb 24, 2013 to March 29, 2013. Posts in a collection contain only one specific event keyword. In our experiment, we used 18 event keywords and finally got 18 sets of microblogs.

DS2. Another dataset DS2 contains posts about specific events. We obtained those events by searching for posts containing several keywords about the specific event. We totally collected 24 events for evaluation.

5.2 Event Tuple Extraction

Table 1. Precision for event type extraction

Technique	Precision
Multinomial Logistic Regression	88.9%
Random Forest	85.6%
Multilayer Perceptron	86.8%
Support Vector Machine	84.4%

5.2.1 Event Type Extraction. In the training step, we randomly select 10 days from DS1, with microblog posts for all query words to construct our training data. Each microblog post collection of a query word in one day is a piece of training data. Thus, we have totally about 180 training data for the 18 queries. We manually label the training data into one of four named-entity category, i.e., location based category, person name-based category, organization-based category and time-based category. We test some machine learning techniques to train the model. We use the trained model to test on the remaining 20 days of microblogs for all queries. Since we do not care much about the recall metric in this task, we use precision to evaluate the method. In our experiment, we use the following methods to calculate the event type, and Multinomial Logistic Regression achieves the best performance, as shown in Table 1.

5.2.2 Event Tuple Extraction. Microblogging topic can usually be divided in two types, on is the event-related topic, the other is the event-independent topic. Therefore, we are not only to evaluate a cluster in the data similarity is high, but also to evaluate whether the microblogging event-related topic. In the experiment of event tuple extraction, we consider two types of precision metrics in the evaluation, i.e., the event cluster precision ***PE*** and the overall cluster precision ***PC***. For each cluster we obtained from the clustering step, we manually check all posts in the cluster. We consider a cluster as a true cluster if more than 80% posts in the cluster are related to the same topic. Further on, if the topic in a true cluster is event related, then the cluster is a true event cluster. Formula (4) and (5) describe the two metrics. ***PE*** is the count of true event clusters divided by the count of all clusters we extracted, and ***PC*** is the count of true clusters divided by the count of all clusters. We also evaluate the ***recall*** value of our metric. Since it is a difficult task to find out all the events in the microblogging collection, the ***recall*** is defined as the number of true event clusters.

$$PE = \frac{\#true\_event\_cluster}{\#all\_cluster} \quad (4)$$

$$PC = \frac{\#true\_cluster}{\#all\_cluster} \quad (5)$$

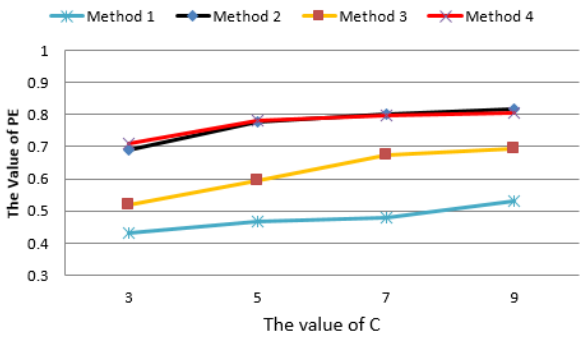

Figure 7. Result of PE for different methods

Figures 7-9 show the results of *PE*, *PC* and *recall*, respectively. We evaluate four methods which are shown in Table 2. The difference between those methods lies in the calculation of similarity in clustering. From Figs. 7-9, we can see that, for the metric *PE*, our methods (Method 4 and Method 2) achieve a synthesis best performance. However, for the metric *PC* and *recall*, the basic Method 1 reaches the highest values. After examining the resultant clusters, we discover that for a basic Method 1, since less restriction is taken on the data when clustering, there are more clusters being aggregated. Thus, the *PC* and *recall* value are higher than methods that based on named entity similarity. However, many clusters in the result cluster set are not event related, which makes Method 1 have the worst performance in terms of *PE*. Note that for a task of event extraction, event related performance is what we need, i.e., we need high *PE* and *recall* values.

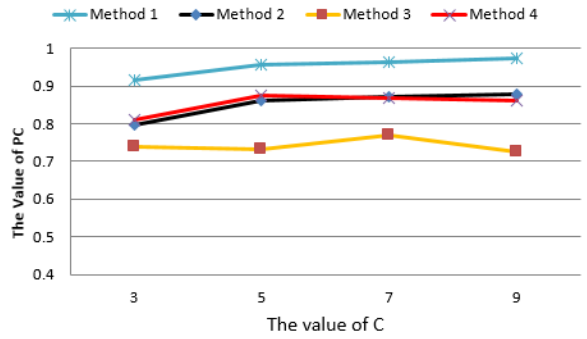

Figure 8. Result of PC for different methods

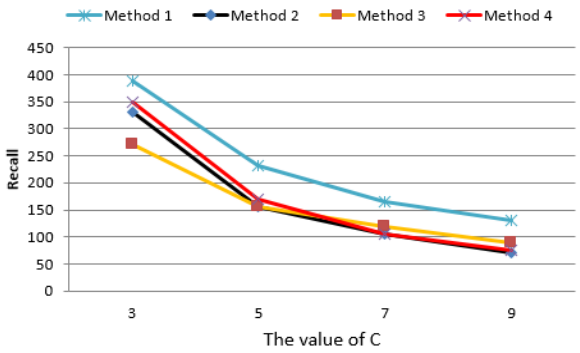

Figure 9. Result of Recall for different methods

Table 2. Methods for Comparison

Method	Description
Method 1	Consider only normal term cosine similarity.
Method 2	Consider only named entity similarity, all category of named entities share the same weight.
Method 3	Consider only named entity similarity, the weight of each named-entity category is decided by their entity frequency in the post collection.
Method 4	Consider only named entity similarity, the weight of each named-entity category is decided by Formula (2)

5.3 Event Tuple Linking

In this experiment, our experimental dataset is DS2. DS2 is composed of 24 real events. Each event has a number of event tuples, containing a total of 752 event tuples. In the experiment, we divide 752 event tuple by day, and then link the event tuples. For each event we obtain from the event tuple linking algorithm, we check all event tuples in the event. We consider an event as a real event if more than 80% event tuples belong to the same event.

In the evaluation, we define the precision ***P*** is the number of real events in the result divided by the total number of events in the result, as shown in (6). In addition, we define the recall ***R*** is the number of event in the dataset divided by total number of events in the result, as shown in (7). Finally, we define ***F1*** as (8).

$$P = \frac{\#real\ event\ in\ result}{\#all\ event\ in\ result} \tag{6}$$

$$R = \frac{\#all\ event\ in\ dataset}{\#all\ event\ in\ result} \tag{7}$$

$$F1 = \frac{2 * P * R}{P + R} \tag{8}$$

Figure 10 shows the results of event tuple linking when the TFIDF and LDA method is used. We define event similarity by (9). We can see that when α is set to 0.8 or 0.9, the method achieves the best performance.

$$Sim(e1, e2) = (1 - \alpha) * TFIDFsim(e1, e2) + \alpha * LDAsim(e1, e2) \tag{9}$$

We consider the time, location, person and other significance information of the event to calculate the similarity between different events. Table 4 shows the results of different methods for similarity measurement. In Table 3, WWW represents the use of the event *when where* and *who* element to calculate the similarity between different event tuples. As we can see from the table, the WWW methods is less effective than the TFIDF and LDA methods, because the event tuple that we extract may contain a small number of microblog posts, which leads to the *when where* element is difficult to extract.

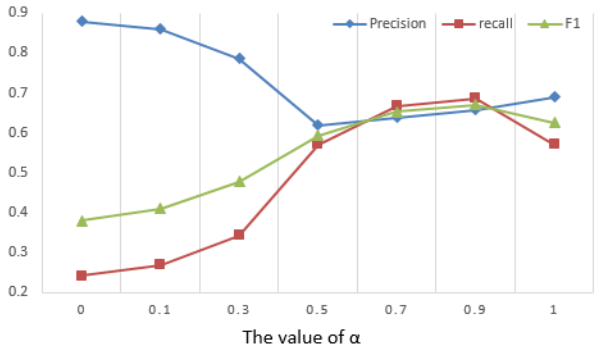

Figure 10. Result of TFIDF-LDA

Table 3. Results of six methods

Method	*P*	*R*	*F*1
WWW	0.947	0.115	0.206
WWW + TFIDF	0.556	0.533	0.544
WWW+TFIDF + LDA	0.875	0.214	0.344
TFIDF + LDA	0.657	0.686	0.671
TFIDF	0.879	0.242	0.380
LDA	0.69	0.571	0.625

Although the TFIDF-LDA method achieves good performance compared with other methods, it is far from meeting our requirements. In the future we will consider other techniques to event tuple link, such as deep learning.

5.4 Event Lifecycle Detection

In this experiment, we also use the dataset DS2. In the baseline method, we use the total number of microblogs of the event tuple in the current time slice to represent the popularity of event tuple, and then use the rate of change of popularity to measure changes in the lifecycle of the event. The formula is shown in (10), where *curPopularity* represents the popularity of the event in current time, and *average* represents the average of the event popularity in the previous **λ** time.

$$changRate = \frac{curPopularity - average}{average} \quad (10)$$

We manually check the division of each stage. If there are multiple identical stages, the division is marked correct if one of the stages is detected correctly. Table 4 shows the precision of the various stages with different values of $\boldsymbol{\lambda}$.

Table 4. Precision of the baseline method

λ	Development	Peek	Recession	Pacification	Average
1	0.647	0.682	0.591	0.45	0.592
2	0.667	0.739	0.65	0.5	0.639
3	0.75	0.739	0.7	0.588	0.694
4	0.625	0.667	0.714	0.619	0.656
5	0.563	0.708	0.667	0.5	0.609

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel method for detecting and monitoring event evolution phase as well as emotional evolution on microblogging platforms. Compared with existing studies, the proposed algorithm presents the lifecycle-based evolutional model for events and emotion changes. With this mechanism, people are able to capture the macro-situation and trends of events and further improve the effectiveness of decision making. In addition, the algorithm can provide detailed 5W1H information for event tuples so that people can view the details of event evolution. We introduce an event-type-driven approach to extract event tuples from raw microblog posts, which can provide a fine-grained description for the extracted events. Finally, we propose a prototype system named EventSys that provides a visualization interface for users to monitor event and emotional evolution.

In the future work, we will study more effective algorithms for event link and lifecycle detection. For example, machine-learning or deep-learning models can be considered to improve the effectiveness of the algorithm. In addition, in future we will develop real-time framework to make the system run on the Web, so that it can continuously crawl microblog posts and perform event evolution analysis.

ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation of China (61672479 and 71273010). Peiquan Jin is the corresponding author of this paper.

REFERENCES

[1] P. Jin, L. Mu, L. Zheng, et al. 2017, News Feature Extraction for Events on Social Network Platforms, *WWW*, pp. 69–78.

[2] L. Zheng, P. Jin, J. Zhao, L. Yue. 2014, A Fine-Grained Approach for Extracting Events on Microblogs. *DEXA*, pp. 275–283

[3] O. Miles, M. Sean, M. Richard, et al. 2014, Real-Time Detection, Tracking, and Monitoring of Automatically Discovered Events in Social Media, *ACL*, pp. 37–42.

[4] H. Cai, Z. Huang, D. Srivastava, et al. 2016, Indexing Evolving Events from Tweet Streams, *ICDE*, pp. 1538–1539.

[5] S. Lin, P. Jin, X. Zhao, L. Yue. 2014, Exploiting temporal information in Web search. *Expert Systems with Applications*, 41(2): 331–341.

[6] J. Huang, M. Peng, H. Wang, et al. 2017, A Probabilistic Method for Emerging Topic Tracking in Microblog Stream, *World Wide Web,* 20(2): 325–350.

[7] P. Lee, L. Lakshmanan, E. Milios. 2013, Event Evolution Tracking from Streaming Social Posts, *arXiv*:1311.5978.

[8] E. Kuzey, J. Vreeken, G. Weikum, 2014, A Fresh Look on Knowledge Bases: Distilling Named Events from News, *CIKM*, 1689–1698.

[9] A. Ritter, O. Etzioni, S. Clark. 2012, Open Domain Event Extraction from Twitter, *KDD*, 1104–1112.

[10] F. Kunneman, A. Bosch, 2016, Open-Domain Extraction of Future Events from Twitter. *Natural Language Engineering*, 22(5): 655–686.

[11] R. Parikh, K. Karlapalem, 2013, ET: Events from Tweets, *WWW*, 613–620.

[12] A. Cui, M. Zhang, Y. Liu, et al. 2012, Discover Breaking Events with Popular Hashtags in Twitter, *CIKM*, 1794–1798.

[13] D. Blei, A. Ng, M. Jordan. 2003, Latent Dirichlet Allocation, *Journal of Machine Learning Research*, 3: 993–1022.

[14] J. Zhao, X. Li, P. Jin. 2012, A Time-Enhanced Topic Clustering Approach for News Web Search, *International Journal of Database Theory and Application*, 5 (4): 1-10

[15] J. Weng, B. Lee. 2011, Event Detection in Twitter, *ICWSM*, pp. 401–408.

[16] X. Zhou, L. Chen. 2014, Event Detection over Twitter Social Media Streams, *The VLDB Journal* ,23(3): 381–400

[17] D. Surian, S. Chawla, 2014, Detection of Spatiotemporal Outlier Events in Social Networks. *Encyclopedia of Social Network Analysis and Mining*, pp. 364–369.

[18] Q. Diao, J. Jiang, 2013, A Unified Model for Topics, Events and Users on Twitter, *EMNLP*, pp. 1869–1879

[19] X. Zhao, P. Jin, L. Yue, 2008. A Novel POS-Based Approach to Chinese News Topic Extraction from Internet. *Proc. of Future Generation Communication and Networking Symposia*, pp. 39-42

[20] D. Zhou, L. Chen, Y. He, 2014, A Simple Bayesian Modelling Approach to Event Extraction from Twitter, *ACL*, pp. 700–705

[21] T. Hua, F. Chen, L. Zhao, et al. 2013, STED: Semi-Supervised Targeted-Interest Event Detection in Twitter. *KDD,* pp. 1466–1469.

[22] L. Li, J. Zheng, J. Wan. 2017, Dynamic Extended Tree Conditioned LSTM-Based Biomedical Event Extraction. *IJDMB*, 17(3): 266–278

[23] V. Natalia, A. Timothy, D. Dmitriy, et al. 2017, Recurrent Neural Network Architectures for Event Extraction from Italian Medical Reports, *AIME*, 198–202

[24] G. Glavas, J. Snajder, 2014, Event Graphs for Information Retrieval and Multi-Document Summarization, *Expert Systems with Applications*, 41: 6904–6916.

[25] V. HaThuc, Y. Mejova, C. Harris, et al. 2009, A Relevance-Based Topic Model for News Event Tracking, *SIGIR*, pp. 764–765.

[26] C. Wei, L. Lee, Y. Chiang, et al. 2014, Exploiting Temporal Characteristics of Features for Effectively Discovering Event Episodes from News Corpora, *JAIST,* 65(3): 621–634.

[27] C. Yang, X. Shi, C. Wei, 2009, Discovering Event Evolution Graphs from News Corpora, *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 39(4): 850–863.

[28] N. Liu, 2009. Topic Detection and Tracking. *Encyclopedia of Database Systems,* pp. 3121–3124

[29] R. Feldman, 2013. Techniques and Applications for Sentiment Analysis. *Communications of the ACM,* 56(4): 82–89.

[30] R. Turney, 2002. Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews. *ACL*, pp. 417–424

[31] S. Brody, N. Elhadad, 2010, An Unsupervised Aspect-Sentiment Model for Online Reviews. *HLT-NAACL*, pp. 804–812

[32] M. Hu, B. Liu, 2004, Mining and Summarizing Customer Reviews. *KDD*, pp. 168– 177

[33] J. Xu, Y. Ding, X. Wang, et al. 2008, Identification of Chinese Finance Text using Machine Learning Method. *SMC*, pp. 455–459

[34] T. Mullen, N. Collier, 2004, Sentiment Analysis using Support Vector Machines with Diverse Information Sources. *EMNLP*, pp. 412–418

[35] L. Zheng, P. Jin, J. Zhao, et al, 2014, Multi-Dimensional Sentiment Analysis for Large-Scale E-commerce Reviews, *DEXA*, pp. 449–463

[36] C. Whitelaw, N. Garg, S. Argamon, 2005, Using Appraisal Groups for Sentiment Analysis. *CIKM*, pp. 625–631

[37] K. Yoon. 2014, Convolutional Neural Networks for Sentence Classification. *EMNLP*, pp. 1746–1751.

[38] X. Zhao, P. Jin, L. Yue. 2015, Discovering Topic Time from Web News, *Information Processing & Management*, 51(6): 869-890

[39] J. Zhao, P. Jin, Q. Zhang, R. Wen, 2014, Exploiting Location Information for Web Search, *Computers in Human Behavior*, 30: 378-388