Frontiers of Computer Science CN 10-1014/TP
https://doi.org/10.1007/s11704-019-8201-6 ISSN 2095-2228
RESEARCH ARTICLE

Event Detection and Evolution in Multi-lingual Social Streams

Yaopeng LIU[1,2], Hao PENG[1,2], Jianxin LI[1,2], Yangqiu SONG[3], Xiong LI[4]

1 Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100083, China
2 State Key Laboratory of Software Development Environment, Beihang University, Beijing 100083, China
3 Department of Computer Science and Engineering, HKUST, Hong Kong, China
4 National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

Front. Comput. Sci., **Just Accepted Manuscript** • 10.1007/s11704-019-8201-6
http://journal.hep.com.cn on February 18, 2019

Front.Comput.Sci.
https://doi.org/10.1007/s11704-019-8201-6

RESEARCH ARTICLE

Event Detection and Evolution in Multi-lingual Social Streams

Yaopeng Liu[1,2], Hao Peng[1,2], Jianxin Li(✉)[1,2], Yangqiu Song[3], Xiong Li[4]

1 Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100083, China
2 State Key Laboratory of Software Development Environment, Beihang University, Beijing 100083, China
3 Department of Computer Science and Engineering, HKUST, Hong Kong, China
4 National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

Abstract Real-life events are emerging and evolving in social and news streams. Recent methods have succeeded in capturing designed features of monolingual events, but lack of interpretability and multi-lingual considerations. To this end, we propose a Multi-Lingual Event Mining model, namely MLEM, to automatically detect events and generate evolution graph in multilingual hybrid-length text streams including English, Chinese, French, German, Russian and Japanese. Specially, we merge the same entities and similar phrases and present multiple similarity measures by incremental word2vec model. We propose an 8-tuple to describe event for correlation analysis and evolution graph generation. We evaluate the MLEM model using a massive human-generated dataset containing real world events. Experimental results show that our new model MLEM outperforms the baseline method both in efficiency and effectiveness.

Keywords Event Detection; Event Evolution; Stream Processing; Multi-lingual Anomaly Detection.

1 Introduction

Real-life events are emerging and evolving in social and news streams, which hold critical materials in real-world occurrences. Monitoring the critical events over time will help policy makers to analyze the whole situation and make right decisions referring to the evolution process. In such cases, it is necessary to determine critical events and monitor them through timeline. Therefore, event detection and evolution have become a focus in current research.

By sending out timely and precise alarms from massive sources like emergent disasters, event detection model can help people take promptly actions to alleviate huge life and economic losses. The problem of event detection has been studied widely in the literature. Bursty detection [1–6], topic models [7–11] and other clustering algorithms [12–23] have succeeded in detecting events from monolingual streams. These techniques are monolingual-oriented, which are lack of real-time and comprehensiveness. Because the foreign startling news needs time to be delivered into domestic major platforms, real-life events need more details and international perspectives. Thus, the first challenge is how to integrate the multilingual post.

In addition to falling short of handling this challenge, the above methods cannot merge phrases appropriately because phrase semantics change over time. For example, *Aba* was the most relavant word to *Wenchuan* in 2012, but more related to *Jiuzhaigou* recently because of *Wenchuan Earthquake* in 2012 and *Jiuzhaigou Earthquake* in 2017. Therefore, the second challenge is how to handle semantic drift problem.

Furthermore, these techniques mainly focus on single source stream either social streams or news streams and did not treat them equally. But which channel discover events earlier depends on circumstances. For example, disastrous events like *Las Vegas Shooting* was perceived promptly in tweets while political affairs like *Trump's Trade Policy* was reported on news media first. In order to ensure the reliability and real-time capability of the event detection model, we collect the multi-lingual text streams from different channels

Received Month dd, yyyy; accepted Month dd, yyyy

E-mail: *lijx@act.buaa.edu.cn*

as the input of the model. So the third challenge is hybrid-length text streams processing.

Previous research on event evolution from news represent event evolution by graph [24–26], but are not fit for short-text or long spanning event evolution. Since the rise of social media, evolution on short-text streams has received much attention, [8, 27] discover an event chain in microblog, but cannot distinguish the evolution patterns such as spliting and merging. [15, 28, 29] represent the evolution by volume on time dimension,which is coarse-grained and lots of significant events are discontinuous in time.

Events are described from different aspects by multi-national official and social media. Observer will know the event development well and make decision wisely and timely based on a “panoramic view” with evolution patterns of it. However, objective reports and full view of opinions from home and abroad are hard to combine together through long time period and extract a graph view of them. The fourth challenge is evolution graph generation. In short, our major challenges are multi-lingual post integration, semantic drift problem, hybrid-length text processing and evolution graph generation.

In seeking to address these challenges, we propose a Multi-Lingual Event Mining (MLEM) model to automatically detect events and generate evolution graph. Firstly, we unified multi-lingual sources into same language, then maintain an evolving phrase graph from text streams, and focus on anomalous phrases when time widow sliding across. Secondly, we utilize incremental word2vec model to merge synonyms at linguistic level and semantic level and solve semantic drift problem. Word2vec models are shallow, two-layer neural networks trained to produce word embeddings. We employ the incrementally learning of hierarchical softmax function [30] rather than the matrix factorization [31] to save time and space complexity. Thirdly, we put attributes time, locations, participants, keywords, emotion, domain, summary and most-related posts together to form an 8-tuple to represent event. We optimize TextRank model by word2vec to generate event summaries. Posts are microblogs, breaking news and forum messages in multiple languages. Fourthly, we introduce word2vec for event filtering to save time in event evolution set generation and adopt phrase subgraphs, locations, participants and summary similarity measurement to build event correlation. Finally, we adopt line clustering to generate event evolution graph incrementally. We have applied the MLEM model to the practical application *RING* available online.[1)]

Here are some highlights of our proposed model MLEM: 1) MLEM can detect events from multi-lingual social streams. 2) Our framework summarizes the information in the stream and for each event as a graph. 3) With the help of phrase graph, we translate event comparing into graph comparing. Thus, efficient graph algorithm can be applied. 4) We present a novel framework to detect and track event in very large, noisy, domain independent and multi-lingual social streams through a very long time.

We present a sketch of our MLEM model for event detection and evolution from multi-lingual text streams in Figure 1. Phrase graph captures strong correlation between phrases. In Figure 1(a), node size represents the phrase frequency, edge thickness indicates the correlation strength. Core nodes and core edges are dark colored in the final step and each event is annotated by core nodes. As time goes on, edge frequency decay and rise while the time window sliding following Eq. 6. In Figure 1(b), event evolution graph grows incrementally through long time period, typical evolution patterns include evolute, merge, split and converge. All events in the evolution graph share same graph, and different graphs may merge at some time point. Key nodes in the phrase graph are made fully connected and each phrase graph is extended with detail phrases, which are light colored, to represent events in the evolution process.

The remainder of this paper is organized as follows. We firstly introduce the related event detection and evolution work in section 2. Then we present our MLEM model and give the parameters configuration in section 3. Thirdly we provide comprehensive comparison experiments on how our model is efficient and effective comparing to the other methods in section 4. Finally, we conclude our work in section 5.

2 Related Work

In this section, we introduce the background of event detection and event evolution.

2.1 Event Detection in Social Streams

Lots of attention is attracted to event detection in social streams. Mathioudakis et al. [1] presented a monitor that identifies events by sharp rise of keywords number in a specified time slice. However, this monitor cannot distinguish different events sharing the same keywords in bursty flow.

[1)] `https://ring.act.buaa.edu.cn/`

Agarwal et al. [14] and Angel et al. [13] both described the social streams as a highly dynamic entity graph, and extracted dense subgraphs as events from the graph. These methods suffer from the loss of single-entity-oriented events or only post attributes like action of the entities.

Yan et al. [9] learn topics by modeling the word co-occurrence patterns in the corpus, to emerge topics inference effectively. But the topics are limited and not suitable for arbitrary event tracking, since there exists future events belong to unknown topics. Nguyen et al. [32] combines content-based features from post text and the propagation of news between viewers to extract and track events from a given social data stream. But it highly depends on friendship networks of users, but it is impossible to obtain whole friendship networks in most cases. Liu et al. [33] detect events based on knowledge base to merge duplicates, which still requires prior knowledge and the time overhead of querying knowledge base can be reduced by using word2vec in our model. None of the above methods addresses the multi-lingual issues.

Lejeune et al. [34] presented a multilingual event extraction system but limited in the epidemic domain. Agerri et al. [35] proposed a multi-lingual framework for event detection, but it relies on extensive language-specific resources and the main contribution is sophisticate pipelines for four specific languages. Our MLEM model employs incremental word2vec model to cover multi languages and introduce semantic information into phrase graph generation.

2.2 Event Tracking in Social Streams

Event Tracking works when events evolute in continuous time. Lin et al. [36] adopt dynamic pseudo relevance feedback to collect related tweets together and generate event storyline. However, it requires a keywords query given by users, which means the effectiveness totally relies on whether these keywords cover the event well. Ge et al. [37] introduced a learning-to-rank model to generate a topically relevant event chronicles for certain period. Above methods need predefined topics or knowledge, which share the same defect with Event Detection based on Topic Model.

Lee et al. [29] modeled the social streams as a dynamic network and extracted (k,d)-core subgraphs to represent events. This model recognizes evolution patterns by tracking the development of subgraphs across time window. This model only monitor events in adjacent windows, which is not applicable for events spanning over a long period.

2.3 Event Evolution with Correlation Building Methods

Yang et al. [25] defined an scoring function to estimate the existence of evolution relationships. This function cannot be applied to long-term spanning events because timestamp measurement is misleading and essential attributes such as summaries, locations, and participants are not taken into consideration. Zhou et al. [38] adopted TFIEF and Temporal Distance Cost factor, Weiler et al. [8] utilized word matrix, Lu et al. [27] used location and participants similarity and Liu et al. [33] add subgraph similarity to measure the event relationships and generate an evolution chain. Most of these methods miss semantic information and evolution patterns and all of them cannot be applied to multi-lingual text streams. Moreover, location and participant have different impacts in a event but these methods treat them equal. We need summary similarity measurement to build event correlation. Most importantly, evolution as a chain is counterintuitive. There exist evolution, splitting, merging and converging in the development of an event.

In our evolution model, we adopt metrics including phrase subgraphs, locations, participants and summaries to evaluate event relationships. The most important part of recognizing relationships is phrase subgraphs similarity and summary similarity measurement. We utilize incremental word2vec model to measure phrase subgraphs similarity and generate reliable summaries for events.

3 Model and Algorithms

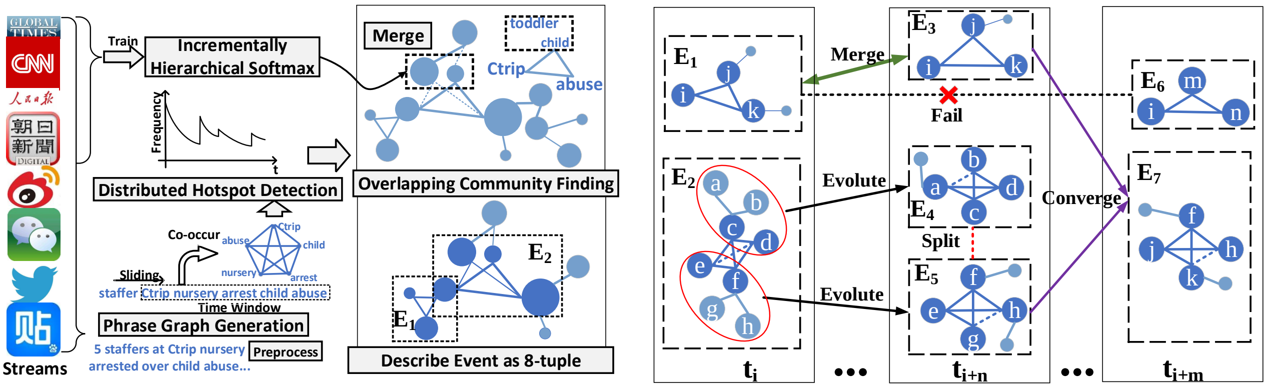
In this section, we introduce our methods for detecting event and generating event evolution graph from multi-lingual text streams, which we call MLEM model.

3.1 Preliminaries

Event Definition. We define an event E as 8-tuple :

$$E =< t; desc; locs; pars; key; emo; dom; posts > \qquad (1)$$

where t is the timestamp when the event emerged in network. $desc$ outlines the event in short. $locs$ is a set of locations the event most related to. $pars$ is a set of people or organizations who mainly participants in the event. key is a set of key phrases of the event. emo is the emotion tendency of related posts, which can be positive, negative or neutral. dom is the domain of the event, including natural disaster, safety accident, environmental pollution, health care, social security, politics, military and entertainment. $posts$ is related posts of



(a) Event Detection

(b) Event Evolution

Fig. 1 Illustration of event detection and evolution from multi-lingual text streams.

the event, with limit size of $\bar{n}$ for each kind of media type after duplication removal.

Event Evolution Graph Definition. Given a latest event E_0, we denote $\mathcal{G}_0(\mathcal{V}, \mathcal{E})$ as the evolution graph of E_0, where $\mathcal{V}$ indicates the set of events $\{E_0, \ldots, E_n\}$ sharing the evolution graph, $\mathcal{E}$ represents the set of relationships between events. For example, $\mathcal{E}$ for the Figure 1(b) is $\{E_1 \sim E_3, E_2 \Rightarrow E_4, E_2 \Rightarrow E_5, E_3 \Rightarrow E_7, E_5 \Rightarrow E_7, E_3 \rightleftharpoons E_5, E_4 \pitchfork E_5\}$. $E_i \sim E_j$ means E_j is merged by E_i. $E_i \Rightarrow E_j$ means E_j is evolved from E_i. $E_i \rightleftharpoons E_j$ means E_i and E_j are going to converge into same event. $E_i \pitchfork E_j$ means E_i and E_j are split from same event. Graph $\mathcal{G}_0$ traces back the whole development of event E_0 along the timeline.

Phrase Graph Definition. We assume that we get edge sequences continuously from text streams, the phrase graph is defined as $G(\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ represent the set of phrases extracted from the text streams and $\mathbb{E}$ is the set of edges corresponding to co-occurrence relationships between phrases in a text sliding window. Specifically, we accept multiple entities or verbs sharing the same meaning on one node in $\mathbb{V}$. The edge weights between the nodes in G will change significantly, as the graph evolves over time. We define the edge weight between node g_i and g_j arrived at t_s as $\mathcal{W}(g_i, g_j, t_s)$.

Phrase Semantic Similarity. Given two phrases w_i and w_j, we define the semantic similarity between them by cosine distance:

$$Sim(w_i, w_j) = \overrightarrow{v_{w_i}} \cdot \overrightarrow{v_{w_j}} \tag{2}$$

where $\overrightarrow{v_w}$ is the unit vector of word w calculated from word2vec model, the normalization of vector v_w is defined as:

$$\overrightarrow{v_w} = \frac{v_w}{\parallel v_w \parallel} \tag{3}$$

where $\parallel v_w \parallel$ is the module of v_w.

Node Similarity. Given two nodes g_i and g_j, we define the similarity between them by the max semantic similarity of phrases on them:

$$Sim(g_i, g_j) = \max_{w_i \in g_i, w_j \in g_j} Sim(w_i, w_j) \tag{4}$$

where $w_i \in g_i$ means phrase w_i is on the node g_i.

3.2 Phrase Graph Generation

Figure 1(a) shows a rough process of phrase graph generation from text streams within the same time sliding window. The size of the time sliding window is set as $\mathcal{T}$ in this paper. We use news media type and social media labels to distinguish the type of languages, for example, HTML tag *tweet-language* of the web pages. Twitter and Weibo are sometimes written for advertising, which introduce lot of noises in multi-lingual text streams. Therefore, we perform multi-lingual standard text processing tasks using the Stanford CoreNLP tool [39]. Specifically, we only retain entities and verbs and texts more than three phrases to reduce noises. In addition, we use a naive bayesian model to train a binary classier and judge whether each post is noise or not.

In our model, we merge the same entities in different languages or expressions like *pistol* and *pistolet* (French), *American* and *U.S.A.*, *American President* and *Trump*. First, the phrase are unified to the same language, then we employ word2vec model to merge multiple synonyms into one node. For each phrase, we go through each node on the phrase graph, if the similarity exceeds threshold Φ, we merge the phrase to the exist node and represent it with the former phrase in lexicographical order. The synonyms merge threshold Φ is tuned to 0.82 in this paper. A phrase sequence is generated for word2vec training and edge connections at the same time. We only retain the sequence from news media because it is informative and objective.

In most cases, people tend to post something meaningful first. For example, the first paragraph or even the first sentence of a report basically sum up the full text, details are available in the rest of the article. Moreover, some people just put hot topics together for advertisement or just comment on them together. Hence simply drawing edges by co-occurrences in one post will introduce a lot of redundancy as detailed information is useless in detection process. It will even generate interfering information and mislead the construction of phrase graph. Therefore, we introduce a text sliding window to draw the weighted edges to distinguish key information and redundant information. If two nodes g_i and g_j co-occur in the same text window, a weighted edge is drawn between them. We define the decay of edge weight W_k through $post_k$ in the current time slice t_s as:

$$W_k(g_i, g_j, d, t_s) = W \cdot 2^{-\Lambda \cdot \frac{d}{l}} \tag{5}$$

where W is a weight constant, Λ is the decay factor for decline rate of information importance, d is the phrase count text window slide away from the beginning of $post_k$, l is the width of window. Since news report are objective and authoritative, we set W as 2 for it and 1 for the others.

For multiple co-occurrence in a single post, we update the weight of this post by maximizing instead of accumulating. After going through all the posts, if there are edges with the same starting point and ending point discriminated by the post ID, we merge them by adding the weights together. We note that the impact of hot topic will slowly wane over time, so the edge weight should not stabilize over a long period of time. In order to model the temporal effect, We introduce a decay factor λ to regulate the rate at which the weight of an edge decays over time. We define the edge weight $\mathcal{W}$ at t_n as:

$$\mathcal{W}(g_i, g_j, t_n) = 2^{-\lambda} \cdot \mathcal{W}(g_i, g_j, t_{n-1}) + \sum_{k=1}^{s_n} \max_{d < post_k} W_k(g_i, g_j, d, t_n) \tag{6}$$

where t_{n-1} is the last adjacent time slice, s_n is the size of posts at time t_n and $d \prec post_k$ means for each position d in $post_k$. Since the graph is assumed to be undirected, the value of $\mathcal{W}(g_i, g_j, t)$ is the same as $\mathcal{W}(g_j, g_i, t)$. Finally, we get phrase graph $G(\mathbb{V}, \mathbb{E})$ defined in 1.

3.3 Event Detection Model

After phrase graph generation, we detect anomalous hot nodes on the phrase graph using the method proposed by Yu et al. [40]. Then we extract subgraph based on the hot phrases. Only retain the anomalous hot nodes and edges between them will miss some events during the detection process which is explained in fig 2. The dark nodes represent hot phrases and the dotted rectangle represent event. Therefore, we retain anomalous hot nodes and nodes meeting the following conditions: node connected to 2 or more anomalous hot nodes and one of its edge weight exceed 8. In this way, we can obtain richer information and ensure the burstiness of events.

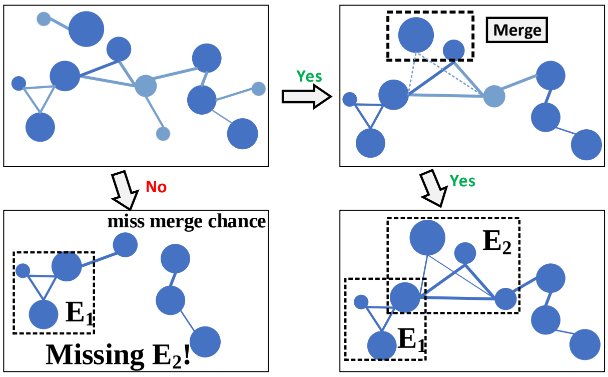


Fig. 2 Illustration of the subgraph extraction.

Moreover, we connect nodes to merge cliques telling the same thing using word2vec. Since the phrase co-occurrence possibility drifts from time to time, we train hierarchical softmax function every 2 hours on old corpus and new corpus generated in 2 through this period incrementally. We connect each node with nodes connected with the other node if cosine distance between two nodes exceeds threshold ϕ. The weights of the newly connected edges are calculated with the help of word semantic similarity. If cosine distance between nodes g_i and g_j exceeds ϕ and we connect g_i, g_h, the weights of new edge is defined as:

$$\mathcal{W}(g_i, g_h, t_n) = Sim(g_i, g_j) \cdot \mathcal{W}(g_j, g_h, t_n) \tag{7}$$

Figure 3 shows an example of the linking process. The solid line represents co-occur relationship, the dotted bidirectional arrow means cosine distance of two nodes exceeds ϕ and the dotted line represents the edge connected through word2vec model. If $cos(d, f)$ exceeds ϕ, we connect b, d , c, d and a, f, merging cliques abc, ade and node f to $abcdef$. For example, nodes d and f could be *pistol* and *handgun*, *toddler* and *child*, and in Figure 3 they're *abuse* and *mistreatment*. Community $abcf$ and ade will be regarded as two events without merging.

Finally, we remove edges with weight less than 1 and use an optimized overlapping community finding algorithm [41] on the subgraph to discover events. We define k-clique as a

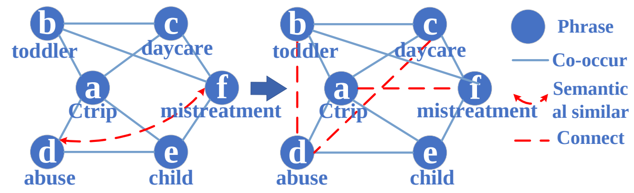


Fig. 3 Illustration of the linking process.

set of fully-connected k nodes and cliques share $k-1$ nodes are called 'adjacent', a community is the largest set the adjacent cliques constitute. We regard each community as an event and represent it by phrases in the community.

After we detect an event, we fill up the 8-tuple to make it understandable. Firstly the time slice when the event is discovered is set as t. Secondly phrases in the community are naturally set as key. Thirdly we track down related posts containing all these keywords by querying post index and put them into set $posts$. To make it applicable to multi-lingual text streams, we index posts with same language and the origin posts are stored in database.

Fourthly we extract $desc$ from $posts$ through our own algorithm. The priority is to set $desc$ as the sentence containing most of the important phrases related to the event. We utilize incremental word2vec to optimize the TextRank model [42] to rank the phrases for event summarization. We use a sliding window to draw edges by co-occurrence relationship within the window to construct an undirected graph. The key idea of TextRank is that the importance of a node depends on how many adjacent nodes point to it, and the weight of the neighboring nodes also affects it. The node weight is defined as:

$$WS(g_i) = (1-\theta) + \theta \sum_{g_j \in I(g_i)} \frac{e_{ji}}{\mid O(g_j) \mid} WS(g_j), \qquad (8)$$

where the $WS(g_i)$ is weight of node g_i, $I(g_i)$ is predecessor set of g_i, $O(g_j)$ is successor set of g_j, e_{ji} is the edge weight of $g_j \rightarrow g_i$, $\theta \in [0,1]$ is damping factor, commonly set as 0.85. The default weight of each node is set to 1 in traditional TextRank model, the successor weights are averaged by the node weight, and they are iterative updated through adjacent relationships. Obviously, a more reasonable way to initialize the node state is to take the influence of each node as an initial state. Our initialization for node g_i is defined as:

$$WS_0(g_i) = b * \sum_{g_j \in In(g_i)} Sim(g_j, g_i) \qquad (9)$$

where $WS_0(g_i)$ is the initial weight of g_i, b is a weight constant, we set b as 2 for node with keyword on and 1 for the others. Thus, Eq 8 is improved to:

$$WS(g_i) = \theta \sum_{g_j \in I(g_i)} \left(\frac{Sim(g_j, g_i)}{\sum_{g_k \in O(g_j)} Sim(g_j, g_k)} + \frac{e_{ji}}{\mid O(g_j) \mid} \right) WS(g_j) + (1-2\theta) \qquad (10)$$

$O(g_i)$ is same as $I(g_i)$ because we generate an undirected graph. We continue iteration until each node weight no longer changes. The phrase weights are used to determine the sentence importance. We set θ as 0.425 in this paper. We split the $posts$ into news set, WeChat article set and the other posts set depends on their type and traverse each subset in authoritative order. The details is displayed in Algorithm 1.

Algorithm 1 Event Summarization.

Input: $posts$

Output: $desc$

```
$maxImp = 0, desc = NULL$;
Split $posts$ into $subsets$ by media type;
Sort $subsets$ in authoritative order;
for each $subset \in subsets$ do
    Construct graph $G_{sub}(V, E)$ for $subset$;
    Calculate weight list $WS_G$ by optimized TextRank;
    for each $p \in subset$ do
        Split $p$ into sentences set $S$ by punctuation;
        for each $s \in S$ do
            $importance = 0$;
            for each phrase $g \in s$ do
                $importance+ = WS[g]$;
            if $importance > maxImp$ then
                $desc = s$;
                $maxImp = importance$;
return $desc$;
```

Then we adopt NLPIR tool to discover $locs$ and $pars$ from the key phrases. NLPIR tool use an automatic identification of entity names based on role tagging to choose participants. According to the role of name recognition, Viterbi algorithm is used to tag the segmentation results, and pattern matching is performed based on character sequence. By closing and opening 16M byte real corpus, the method recall achieves up to 98%. The most frequently mentioned locations are put into set $locs$, people and organizations are put into set $pars$. Specifically, if the frequency difference between top mentioned locations and most mentioned location is less than 20%, we put all of them into $locs$. We extract $locs$ and $pars$ from key phrases, if fails, we will go through related posts, count the locations and participants and fill $locs$ and $pars$ again. Finally, we employ SVM model based on cell thesaurus for domain classification and Bayesian model improved by emoticons to classify sentiments [43].

We describe events from eight aspects, which is concise and easy to understand. An example is showed in Table 1, which is about Child abuse in Ctrip daycare in Nov.08.2017.

Table 1 Event Description Case

Tuple	Tuple description
t	2017/11/08 14:20
$desc$	Child abuse in Ctrip daycare.
$locs$	Shanghai
$pars$	Jie Sun
key	Child, abuse, Ctrip, daycare
emo	negative
dom	social security
$posts$	Maltreatment of toddlers in Ctrip daycare in Changning District, Shanghai # Child abuse scandal $\cdots$

3.4 Event Correlation Building Method

Determine the relationship between two events is a major challenge in building an evolution map of an event. Given two events E_i and E_j represented by the 8-tuple. We define event similarity score function as:

$$\begin{aligned} Sim(E_i, E_j) = \alpha \cdot Sim_{subgraphs}(E_i, E_j) + \beta \cdot Sim_{locs}(E_i, E_j) \\ + \gamma \cdot Sim_{pars}(E_i, E_j) + \delta \cdot Sim_{desc}(E_i, E_j), \end{aligned} \tag{11}$$

where $Sim_{subgraphs}$, Sim_{locs}, Sim_{pars} and Sim_{desc} denote similarity measures of phrase subgraphs, locations, participants and summaries respectively. $\alpha, \beta, \gamma, \delta$ are weight coefficients of them subjected to $\alpha + \beta + \gamma + \delta = 1$.

The event is multifaceted but limited sides have clues. Therefore, we use $Sim_{subgraphs}$ as a significant feature rather than global text similarity. Firstly we form a phrase graph for the event. We get top 6 frequent key phrases from *key*. We set detail phrases as top 8 frequent phrases apart from the key phrases in *posts* co-occur with key phrases in a text sliding window. Then we merge multiple synonyms through the method in 2. We connect each phrase pair in a text sliding window and make key phrases fully connected after that. Figure 1(b) shows the final graph for each event.

Moreover, we consider each three connected nodes with at least one key node as a subgraph, and calculate each subgraph pair Sim_T's similarity. We set three nodes as subgraph is because intuitively, event can be represented by three phrases like *somebody did something* or *something happened somewhere*. We represent each phrase by a vector using word2vec, and calculate Sim_T by cosine similarity of incenters for subgraph pair. The incenter of a triangle is the crossover point of three interior angle bisector. We use a triple vector $T_i =< \overrightarrow{v_{i,1}}, \overrightarrow{v_{i,2}}, \overrightarrow{v_{i,3}} >$ to represent each subgraph, where $\overrightarrow{v_{i,j}}$ denotes the unit vector of phrase j in T_i, and the incenter $v_{i,incenter}$ is defined as follows:

$$v_{i,incenter} = \frac{z \cdot \overrightarrow{v_{i,1}} + x \cdot \overrightarrow{v_{i,3}} + y \cdot \overrightarrow{v_{i,1}}}{x + y + z}, where \begin{cases} x = |\overrightarrow{v_{i,1}} - \overrightarrow{v_{i,2}}|, \\ y = |\overrightarrow{v_{i,1}} - \overrightarrow{v_{i,3}}|, \\ z = |\overrightarrow{v_{i,2}} - \overrightarrow{v_{i,3}}|, \end{cases} \tag{12}$$

Based on Eq (12), the incenter vector is an unit vector, so we define $Sim_{subgraphs}$ as the maximum similarity among all subgraph pairs between two events:

$$Sim_{subgraphs}(E_i, E_j) = \max_{i \in [1,n], j \in [1,m]} v_{i,incenter} \cdot v_{j,incenter}, \tag{13}$$

where n and m are the count of subgraphs in each event. Reports from different aspects focus on different places and different people, which is needed to be distinguished in the evolution graph. The simple location similarity cannot meet the demand. Motived by this, we adopt the weighted Jaccard similarity which is applied in combination with the signal based similarity. The weighted Jaccard similarity was first proposed by Ioffe et al [44]. For our problem, the weight of our locations is the frequency in each *posts*. Thus, for a given event pair, it measures the similarity of the places each event focus on and their importances. We define Sim_{locs} and Sim_{pars} as:

$$\begin{cases} Sim_{locs}(E_i, E_j) = \dfrac{\sum_c \min(FL_c^i, FL_c^j)}{\sum_c \max(FL_c^i, FL_c^j)} \\ Sim_{pars}(E_i, E_j) = \dfrac{\sum_p \min(FP_p^i, FP_p^j)}{\sum_p \max(FP_p^i, FP_p^j)}, \end{cases} \tag{14}$$

where FL_c^i donates the frequency of location c in $posts_i$, FP_p^i donates the frequency of participant p in $posts_i$. We represent each $desc$ as a weighted phrase vector $d_i =< w_{i,1}, w_{i,2}, \cdots, w_{i,N} >$. $w_{i,j}$ denotes the weight of phrase j in d_i and N is the vocabulary size. Sim_{desc} is defined as follows:

$$Sim_{desc}(E_i, E_j) = \frac{\sum_{k=1}^{N} w_{i,k} \cdot w_{j,k}}{\sqrt{\sum_{k=1}^{N} (w_{i,k})^2} \cdot \sqrt{\sum_{k=1}^{N} (w_{j,k})^2}}, \tag{15}$$

where key phrases and entities' weight is 2, non-exist phrases' weight is 0 and other phrases' weight is 1. The reason we take empirical value rather than frequency as weight is that summary of article is unlikely to obtain duplicate words.

When we analyze the KEM model, we find that some events which should have evolution relationship failed evolution because their similarity is lower than ϵ but phrase subgraphs similarity is high enough. Then we find that events evolute in the useful subgraph part, it already contains clue of the event. Thus, after we obtain $Sim_{subgraphs}(E_i, E_j)$ or $Sim(E_i, E_j)$, we define that E_j is evolved from E_i when $Sim_{subgraphs}(E_i, E_j) > \varepsilon$ and $t_i < t_j$ or $Sim(E_i, E_j) > \epsilon$, i.e.,

$$E_i \rightarrow E_j \;\; if \begin{cases} Sim_{subgraphs}(E_i, E_j) > \varepsilon \;\; or \;\; Sim(E_i, E_j) > \epsilon \\ t_i < t_j. \end{cases} \tag{16}$$

We tune ε to 0.25 and ϵ to 0.45 in this work.

3.5 Event Evolution Graph Generation

After relationships are discovered, we focus on generating the evolution graphs incrementally. In this work, we adopt line clustering rather than point clustering because events may evolute into a very different event. It extends the evolution graph incrementally using less time and space. Specifically, we get an event evolution set and merge it into the graph of the most-related event generated before, which is clustered from point to point rather than clustering from one point.

Firstly, we employ two-layer filtering method introduced in [27] to save computational time in the following steps. We get a candidate events set CS with maximum size 20 for input event E_0 after filtering. We adopt semantic distance filtering to further reduce the computing time. Each event E_i in set CS and E_0 owns a vector set $VS_i =< v_{i,1}, v_{i,2}, \cdots, v_{i,m} >$, where $v_{i,j}$ is word vector of the j^{th} key phrase in E_i, m is the size of key phrases. We define a weighted average vector $V_{i,E}$ to discover irrelevant events and is computed as follows (i.e. the AverageVector method at line 2 and 7 in Algorithm 2):

$$V_{i,E} = \frac{\sum_{k=1}^{m} f_{i,k} \cdot v_{i,k}}{\sum_{k=1}^{m} f_{i,k}}, \tag{17}$$

where $f_{i,j}$ denotes the frequency of key phrase j in related posts of E_i. When cosine distance between $V_{i,E}$ and $V_{0,E}$ below threshold φ, we drop the event E_i in set CS. We tune φ to 0.1 in this paper to enhance best performance.

Then we head to find all events E' for input event E_0 meeting the condition: $E' \rightarrow E_0$, making a set of available events: S_E. We get evolution graph G of event with the maximum similarity in S_E by querying the event index, reorder S_E by time ascending order and insert E_0 at the end of S_E. Algorithm 2 (i.e. the EESG method at line 1 in Algorithm 3) illustrates the method to generate event evolution set for given event E_0. Set S_E is the available event set for input event, Graph G is the evolution graph.

Finally, we put events from S_E to the graph G, draw edges between adjacent events in S_E if no path exists between them, and discover relationships between events in G. We define split relationship as different events within a day directly evolved from same event, merge relationship as same events detected in continuous time span and converge relationship as different events directly evolved to same event. We tune ϵ' and ε' as 0.8 in the experiment to merge same events. Since breaking news or buzz topic can keep high heat through a long period of time, we merge events back to the earliest one and represent its t as a time span striding across multiple time slice.

Algorithm 2 Event Evolution Set Generation.

```
Input: Event $E_0$
Output: Set $S_E$, Graph $G$
 1: $sim_e^{max} = sim_{sub}^{max} = 0, pos_{max} = 1$;
 2: $V_{0,E} = AverageVector(E_0)$;
 3: Get events set $CS$ by searching event index;
 4: Drop events in set $CS$ by two-layer filtering;
 5: for $i = 1$; $i \le$size($CS$); $i++$ do
 6:     $E_i = CS[i]$;
 7:     $V_{i,E} = AverageVector(E_i)$;
 8:     if $cos(V_{0,E}, V_{i,E}) < \varphi$ then
 9:         continue;                ▹ semantic distance filtering
10:     $sim_e = Sim(E, E_i)$;
11:     $sim_{sub} = Sim_{subgraphs}(E, E_i)$;
12:     if $sim_e > \epsilon$ or $sim_{sub} > \varepsilon$ then
13:         if $sim_e^{max} \le sim_e$ and $sim_{sub}^{max} \le sim_{sub}$ then
14:             $sim_e^{max} = sim_e, sim_{sub}^{max} = sim_{sub}$;
15:             $pos_{max}$ =size($S_E$) + 1;
16:         $S_E$.add($E_i$);
17: Get $G$ of $S_E[pos_{max}]$ by querying event index;
18: Sort $S_E$ in time ascending order;
19: $S_E$.add($E_0$);
20: return $S_E, G$;
```

Algorithm 3 illustrates the method to generate event evolution graph for given event E_0. HashMap E_G is the set of events in graph G, 2D Array R_G is the set of event relationships in graph G, $E_i \rightsquigarrow E_j$ means there is a path between E_i and E_j, t' and t_j are the timestamps of E' and $S_E[j]$.

Figure 4 shows an example of evolution graph, it could be completed by putting following S_E in a graph: $\{E_1, E_2\}$,$\{E_1, E_3\}$,$\{E_1, E_2, E_4\}$,$\{E_2, E_5\}$,$\{E_3, E_6\}$,$\{E_5, E_7\}$, $\{E_4, E_8\}$,$\{E_5, E_7, E_9\}$. The solid line represents evolutionary relationship, the dotted line represents split relationship (red) and converge relationship (purple), the bidirectional arrow means latter is merged by former. The darkness of each block represents event heat. Since events are not observed within a day, there is no relationship between E_2 and E_3, E_4 and E_5. There exists multiple split or converge relationships, for example, *Jiuzhaigou earthquake* is split into *Counter-Rumour*, *Traffic Control*, *Victims Statistics* and *Disaster Relief* and fi-

Algorithm 3 Event Evolution Graph Generation.

```
Input: $E_0$
Output: Graph $G(E_G, R_G)$
$S_E, G(E_G, R_G) = EESG(E_0)$, $j = 1$;
repeat
    $sim_e = Sim(S_E[j-1], S_E[j])$;
    $sim_{sub} = Sim_{subgraphs}(S_E[j-1], S_E[j])$;
    if $sim_e > \epsilon'$ or $sim_{sub} > \varepsilon'$ then
        Merge $S_E[j]$ to $S_E[j-1]$;
        $S_E$.remove($S_E[j]$);
    else
        if $S_E[j] \notin E_G$ then
            $E_G$.add($S_E[j]$);
        if never $S_E[j-1] \rightsquigarrow S_E[j]$ then
            $R_G[E_G[S_E[j-1]]][E_G[S_E[j]]] = evolute$;
            if $\exists E' : E_G[S_E[j-1]] \Rightarrow E'$ and $| t' - t_j |< 1$ day then
                $R_G[E_G[S_E[j]]][E'] = split$;
            if $\exists E' : E' \Rightarrow E_G[S_E[j]]$ then
                $R_G[E_G[S_E[j-1]]][E'] = converge$;
        $j++$;
until $j > size(S_E)$
return $G(E_G, R_G)$;
```

nally they converge into *Public Memorial Ceremony*.

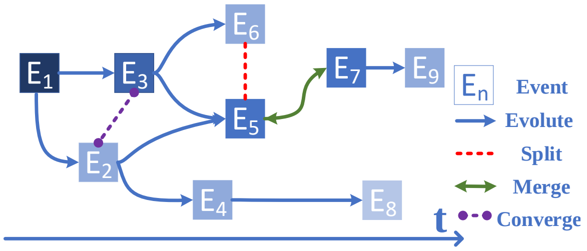


Fig. 4 Illustration of the subgraph extraction.

4 Experiments

In this section, we present rich experiments to verify the effectiveness and efficiency of our MLEM model. We first describe the dataset we adopt for our experiments and the parameters setting, then we evaluate the speed, quality and time delay of event detection model. Finally, we measure the efficiency and effectiveness of our event evolution method and give a case study.

Dataset: We have been collecting multi-lingual text data and detecting events since Feb.12.2016. Raw data including Twitter, Weibo, WeChat, worldwide Publishing House and Forum is stored in HBase and indexed by Elasticsearch. Representative media is shown in Figure 1(a). Until now there has been about 6.4 billion tweets and weibos, 7.8 million news, 15.6 million forum messages collected by crawlers through API or web page, from which about 2.1 million events have been detected.

Parameters: The parameters $W(0)$, λ, l, α, β ,γ ,δ , ϵ, ε, ϵ', ε', Φ, ϕ, φ, n_hashes, n_tables, $n\_neighbours$ used in this paper are listed in Table 2.

Table 2 Parameters Setting

Parameter	Default	Description
$\bar{n}$	5	size of each kind of posts
W	1,2	coefficient of edge weight
$\mathcal{T}$	600s	time window size
λ	2	decay factor over time
Λ	0.05	decay factor over post
l	10	width of the text sliding window
θ	0.425	damping factor
α	0.55	coefficient of subgraphs similarity
β	0.1	coefficient of location similarity
γ	0.15	coefficient of participant similarity
δ	0.2	coefficient of summary similarity
ϵ	0.45	threshold of evolution relationship
ε	0.25	threshold of evolution relationship
ϵ', ε'	0.8	threshold for merging events
Φ	0.82	threshold of merging synonyms
ϕ	0.78	threshold of merging phrases
φ	0.1	threshold of semantic distance filter
n_hashes	5	number of hashes in LSH
n_tables	5	number of hash tables in LSH
$n\_neighbours$	20	number of neighbors to find in LSH

4.1 Event Detection

In this section we evaluate the efficiency and effectiveness of our event detection model. 1 million multi-channel multi-lingual posts in one day are randomly picked out from our post database and treated as the input of Event Detection Model.

Baseline. Works like [27] firstly detect trending keywords using [40] and then adopt overlapping community detection method [45] to discover events from these trending keywords in a time slice. We refer to the baseline method as EECM in this article. Works like [28] detect events by detecting volume peaks of hashtag over time in social streams. We refer to this baseline method as HashtagPeaks. Works like [32] combines content-based features and the propagation of news to detect events. We refer to this baseline method as RTED. We implement it with treating forwarding network as friendship network. However, the above baseline methods are not designed for event evolution because the internal structure of detected events is missing. Works like [29] modeled the social streams as a dynamic network and detect events by extracting (k,d)-core subgraphs. We refer to this baseline method as eTrack.

4.1.1 Efficiency

In this part we evaluate the efficiency of our model MLEM against our preliminary Knowledge-based Event Mining model(KEM) and baseline method EECM [27]. We log the graph size when detecting events, we see that the graph is very large with up to 1 million nodes and 453,173 on average. The standard text processing time is also counted as detection time. Figure 5 shows the comparison of event count in each time slice by the three methods. Figure 6 shows the comparison of average detection time for each detected event by each method.

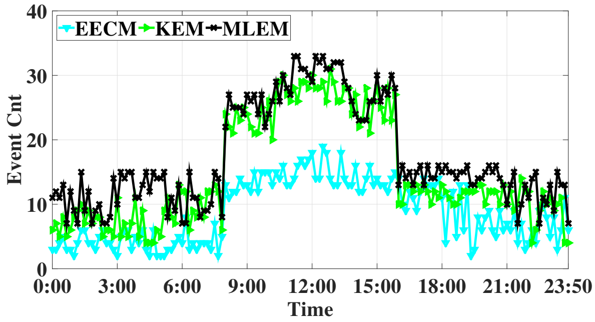


Fig. 5 Comparison of detected event quantity

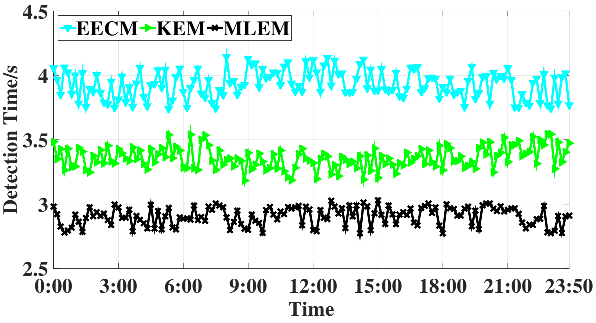


Fig. 6 Comparison of detection time

We observe that the events stream peak occurs roughly between eleven and half past thirteen, corresponding to people's active period within a day. The average whole detection time for each time slice is 35s with EECM, 49.5s with KEM and 50.7 with MLEM. Meanwhile, the average count of events is 8.8, 14.8 and 17.4 on each method, the average detection time for each event is 3.93s, 3.35s and 2.90s on each method. In other words, the detection model increases the event count by 44.8% and improves the computing speed by 35.27% comparing to EECM. The main reason is EECM connects all words in a single long report, which makes the word graph much complicated and redundant. Moreover, the clique EECM detected contains too many words, so lots of computation time is wasted when fail retrieving the posts with too many interference. The improvement compared to KEM model is because of the introduction of multi-lingual sources make the foreign events be sniffed earlier and dereferencing of knowledge base save the time.

4.1.2 Effectiveness

In this part we compare each method mentioned before in terms of detection effectiveness.

Ground Truth. We extract events from international authoritative news articles in September, 2017. These news all have enough related posts obviously and news title is set as summary for each event.

Evaluation. We use the standard metrics Precision(P), Recall(R), F1-Measure(F1) and average time delay ΔT to quantize the effectiveness of our model. They are calculated as follows:

$$\begin{cases} P = \dfrac{|G \cap C|}{|C|} \\ R = \dfrac{|G \cap C|}{|G|} \\ F1 = \dfrac{|2 \cdot PR|}{|P + R|} \\ \Delta T = \dfrac{\sum_{E_i \in G \cap C}(T_{i,detect} - T_{i,emerge})}{|G \cap C|}, \end{cases} \tag{18}$$

where G is the set of events in the ground truth dataset, C is the set of events detected, $T_{i,emerge}$ is when event E_i takes place and $T_{i,detect}$ is when event E_i is detected.

Table 3 illustrates the P,R,F1,ΔT of each method over the dataset. The difference of recalls is much larger than precisions, it is mainly because the merging of same entities in 2 and edge connection in 3 lift heat of each node on the graph to an anomalous level so the recall is improved also. The difference between time delay is because the other methods is unable to process multi-lingual sources and , it takes time to report foreign news in major language. Furthermore, the weakness for processing long text lead to the gap of effectiveness. It is explicit that our method achieve superior effectiveness over other methods.

Table 3 Detection effectiveness results

Method	P	R	F1	ΔT/min
HashtagPeaks	0.4810	0.3367	0.3961	37.01
RTED	0.5761	0.3533	0.4380	39.27
eTrack	0.5684	0.3600	0.4408	40.09
EECM	0.5833	0.3733	0.4553	36.46
KEM	0.6133	0.6767	0.6434	14.23
MLEM	0.6891	0.7833	0.7332	10.14

4.2 Event Evolution

In this section we evaluate the efficiency and effectiveness of our event evolution model and present a case study.

4.2.1 Efficiency

In this part we evaluate the efficiency of our algorithm. 1000 events are randomly picked out from our event database and treated as the input of the evolution model.

Figure 7 shows the distribution of event evolution graph size produced by our algorithm. We can see that 53% events doesn't maintain a graph, which means they have no previous related events. It is because events detected in the social media develop with quick perception and extinction. Moreover, 67% of the rest have evolution graphs with size between 2 and 10, which means that evolving graph have no more than 10 parts commonly. In particular, we check graphs with size larger than 20 and find they are correspond to events of great influence like *DPRK Nuclear Crisis*, *Terrorist Attacks in Paris*, *Hurricane Harvey* et al.

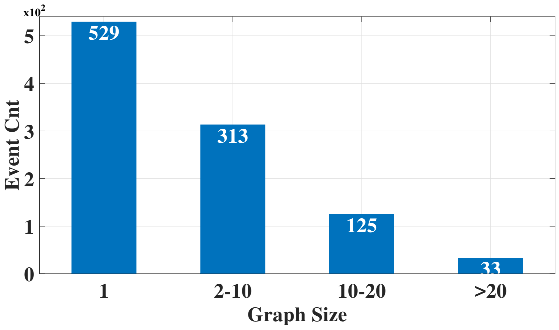


Fig. 7 Distribution of evolution graph size

Figure 8 indicates the comparison of average time spent on event evolution set generation step between our algorithm and algorithm in KEM and EECM, which do not have semantic distance filtering step. We can see that the average time of all 1000 events is 2.18s with filtering and 4.32s without filtering. When only consider events whose S_E size are greater than 2, the time rises to 10.73s and 18.12s. We can see that with the increase of S_E size, the time ratio decrease since candidate set size increase faster than the size of irrelevant events set. On average, the semantic distance filtering step improves the calculation speed by 98.17%.

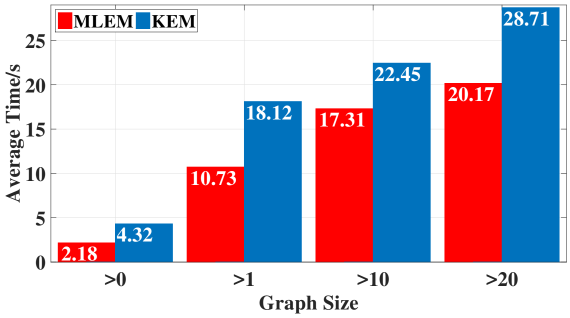


Fig. 8 Comparison of average time on evolution graph generation

4.2.2 Effectiveness

In this part we compare our algorithm with EECM and ground truth generated from news to evaluate the effectiveness.

Ground Truth. We manually choose 300 input events from the 1000 events mentioned before. These events all have obvious evolution processes in real world and corresponding reports of each evolution part of them are easy to find. We treat news title as summary for each evolution part and manually put them on an evolution graph for each event. We labeled 48681 posts for event ground truth. Events are labeled by 10 specialists, each event is reviewed by at least 2 specialists, when there is a divergence of views, another specialist is asked to label the event and the most voted one is set for each event finally.

Baseline. Works like [46] combines content similarity and temporal proximity to measure the relationships between events. We refer to this baseline method as CSTP. eTrack [29] track the development of events in adjacent windows. EECM [27] use the combination of multiple features to measure the relationships between two events. Therefore we adopt CSTP, eTrack, EECM and KEM as baselines. In CSTP, the similarity between events is computed as follows:

$$Sim(E_i, E_j) = Tp(E_i, E_j) \cdot Sim_{posts}(posts_i, posts_j), \quad (19)$$

where $Tp(E_i, E_j)$ is the temporal proximity between E_i and E_j and it's given by:

$$Tp(E_i, E_j) = e^{-\frac{\chi |t_i - t_j|}{T}}, \quad (20)$$

where T is the time distance between the earliest and the latest event in our system. χ is the time decay factor and is set as 1 in this experiment. $Sim_{posts}(posts_i, posts_j)$ is related posts' similarity between E_i and E_j and it is given by:

$$Sim_{posts}(posts_i, posts_j) = \frac{\sum_{p_k \in posts_i} \sum_{p_l \in posts_j} cos\_sim(p_k, p_l)}{|posts_i| \cdot |posts_j|}, \tag{21}$$

where $cos\_sim(p_i, p_j)$ is calculated as same as Eq (15) except the $w_{i,j}$ donates the frequency of term j in p_i.

In EECM, the similarity between events is computed as follows:

$$\begin{aligned} Sim(E_i, E_j) &= \alpha \cdot Sim_{posts}(posts_i, posts_j) \\ &+ \beta \cdot Sim_{locs}(locs_i, locs_j) + \gamma \cdot Sim_{pars}(pars_i, pars_j). \end{aligned} \tag{22}$$

In KEM, the similarity between events is computed as follows:

$$\begin{aligned} Sim(E_i, E_j) &= \alpha \cdot Sim_{subgraphs}(key_i, posts_i, key_j, posts_j) \\ &+ \beta \cdot Sim_{loc}(loc_i, loc_j) + \gamma \cdot Sim_{par}(par_i, par_j), \end{aligned} \tag{23}$$

where $Sim_{subgraphs}$ is the phrase subgraphs' similarity between E_i and E_j and it is given by:

$$Sim_{subgraphs} = \max_{i \in 1,n, j \in 1,m} Sim_T(T_i, T_j), \tag{24}$$

where $Sim_T(T_i, T_j)$ is the similarity of each phrase subgraph pair and is calculated by cosine distance of average vector of each subgraph.

Both of them calculated Sim_{loc} and Sim_{par} as follows:

$$Sim_{loc}(loc_i, loc_j) = \begin{cases} 1 & if\ \ loc_i\ \ equals\ \ loc_j; \\ 0 & otherwise. \end{cases} \tag{25}$$

and

$$Sim_{par}(par_i, par_j) = \frac{par_i \cap par_j}{par_i \cup par_j}. \tag{26}$$

Meanwhile, EECM and KEM define evolution relationship as $Sim(E_i, E_j) > \epsilon$ and $t_i < t_j$. Specially, they adopt point clustering which means the whole evolution chain is generated by a single event, so all the events on the chain show high similarity with each other.

Evaluation. We use P, R, F1 mentioned before to evaluate the effectiveness of our algorithm. In this evaluation, G is the set of events in the ground truth dataset, C is the set of events in output result for Eq (18).

Table 4 shows the P, R, F1 of MLEM with different settings of parameters α,β, γ, δ, KEM, EECM with best combination of parameters, eTrack and and CSTP comparing to the ground truth, the parameters have been optimized for improving the results for each baseline method. To select the best combination of parameters, first we initialize all parameters with random value except first parameter, and find the $parameter_1$ with best performance, then we set first parameter as $parameter_1$, then we fix all parameters except second parameter... when all parameters set, we continue finding best $parameter_1$, $parameter_2$... until all parameters' difference from their previous value is less than 0.1.

Table 4 Evolution effectiveness results

Method	α	β	γ	δ	P	R	F1
MLEM	1	0.0	0.0	0.0	0.5228	0.3228	0.3992
	0.8	0.05	0.05	0.0	0.4470	0.3041	0.3619
	0.7	0.1	0.1	0.0	0.5889	0.5298	0.5578
	0.6	0.1	0.1	0.2	0.5758	0.5361	0.5552
	0.55	0.15	0.15	0.15	0.6177	0.6332	0.6254
	0.55	0.1	0.15	0.2	**0.6294**	0.7241	**0.6735**
	0.5	0.1	0.15	0.25	0.5970	**0.7429**	0.6620
	0.4	0.15	0.2	0.25	0.5194	0.6301	0.5694
	0.4	0.15	0.25	0.2	0.5090	0.6238	0.5606
KEM	0.7	0.1	0.2	-	0.5896	0.5489	0.5685
EECM	0.7	0.1	0.2	-	0.5441	0.4451	0.4897
eTrack	-	-	-	-	0.5665	0.4138	0.4783
CSTP	-	-	-	-	0.5283	0.3511	0.4218

For our method MLEM, we find that the setting $\alpha = 0.5, \beta = 0.1, \gamma = 0.15, \delta = 0.25$ has the highest recall and the setting $\alpha = 0.55, \beta = 0.1, \gamma = 0.15, \delta = 0.2$, which is set as default setting, has the highest precision and F1 score, and outperforms other methods on every metrics.

Remarkably, we find the recall for other methods in our experiment are much lower than MLEM, we check the ground truth data and find out that the earliest and the latest events in the evolution graph are not necessarily similar to each other. For example, a disastrous event may evolute to a political event since government officials may be accountable for the accident. But the other methods consider them as irrelevant events because it doesn't reach the evolution threshold. In our method, we get the old evolution graph already constructed before by the most similar event and put our candidate events into it, so events do not need to be similar enough to be in one graph, which leads to the increase of recall.

Moreover, the other methods only take single location into account, which lose evolution part in evolution graphs associated with multiple locations. In addition, we find that the recall of CSTP is much lower than other methods, we check the ground truth and find out that some events in the evolution graph have long time distance, which is not accepted as relationship by CSTP.

It is worth mentioning that we applied the MLEM to practical application RING and found that the promptness and effect is satisfactory. This is another proof that our MLEM model has good accuracy and strong robustness on event detection and evolution task.

Table 5 Case Study

Ground Truth	MLEM	EECM
2016/03/03 Korean media said North Korea launched several short-range missiles.	2016/03/03 Korean media said North Korea launched several short-range missiles.	2016/03/03 Korean media said North Korea launched several short-range missiles.
2016/03/12 The US and South Korea held Ssangyong training, North Korea intended to pre emptive retaliation.	2016/03/12 The US and South Korea held Ssangyong training, North Korea intended to pre emptive retaliation.	
. . .	. . .	
2016/07/19 07:00 North Korea launches 3 ballistic missiles to the east of the peninsula in Huangzhou.		
2016/07/19 10:40 DPRK launched 3 missiles this morning, possibly protesting South Korea's decision to deploy Sade.	2016/07/19 07:00~10:40 DPRK launched 3 missiles this morning, possibly protesting South Korea's decision to deploy Sade.	
2017/02/12 North Korea launches a flying object, South Korean military analysis suspected to be Musudan missile.	2017/02/12 North Korea launches a flying object, South Korean military analysis suspected to be Musudan missile.	2017/02/12 North Korea launches a flying object, South Korean military analysis suspected to be Musudan missile.
2017/08/29 North Korea launched a missile this morning, flying across Japan.	2017/08/29 North Korea launched a missile this morning, flying across Japan.	
2017/09/15 Japan has strongly condemned the North Korean missile launch.(*translated from Japanese*)	2017/09/15 **Japan has strongly condemned the North Korean missile launch.**	2017/09/05 *South Korean Navy holds live ammunition shooting exercises to strengthen maritime combat capability.*
2017/09/15 South Korean army launched a ballistic missile against North Korea.	2017/09/15 **South Korean army launched a ballistic missile against North Korea.**	2017/09/15 South Korean army launched a ballistic missile against North Korea.
2017/09/15 South Korea's basaltic missile launch abnormal, fall into the sea.	2017/09/15 South Korea's basaltic missile launch abnormal, fall into the sea.	2017/09/15 South Korea's basaltic missile launch abnormal, fall into the sea.

Case study. Table 5 shows the comparison between our method and EECM model based on an example about *North Korea missile problem*. The ground truth describes it with several evolution parts and we represent some unimportant part as ellipsis. For EECM, it fails discovering some evolution parts for reasons like only accept single location, unable to merge same entities and point clustering. Four locations occur in this scene: North Korea, South Korea, Japan and U.S., the first two is main location but EECM only pick one, so the similarity between the second event and the last is calculated inaccurately and EECM miss the second event. EECM model treats DPRK and North Korea as different entities and fail evolution for the third event. Furthermore, EECM gets evolution chain most related to South Korea but the whole evolution graph is mainly on military interaction between North Korea and South Korea, which leads to evolution loss and mistake like the italic event. For our method, it covers all the evolution parts of this long-term event and discover splitting relationship between the bold events, the former of which is detected from Japan media. In addition, the merged event's t is underlined, representing by time span. This case explains our method's advantage on discovering evolution patterns and high accuracy over the baseline method EECM.

5 Conclusion

In this paper, we propose a novel model called MLEM for multi-lingual event detection and evolution graph generation. We introduce incremental word2vec to merge synonyms in detection process. We combine the similarity measures of phrase subgraphs, locations, participants, and summary to evaluate the relationships among events. We adopt two-layer filtering and semantic distance filtering to get the related events for each given event to save calculation time. Experiments show the high performance of our MLEM model in efficiency and effectiveness. For efficiency, our detection model improves computing speed by 35.27% for each event, and our evolution algorithm can generate an event evolution graph for a given event in less than 2.5s on average, which is 98% speedup against the method without semantic distance filtering. For effectiveness, our MLEM model outperforms baseline method EECM and preliminary model KEM on both precision, recall and time delay. The case study shows that the evolution graph produced by our MLEM model covers ground truth and discovers evolution patterns well.

When tracking events in post stream which is very large, noisy, domain independent and multi-lingual, through a very long time such that lots of semantic drift occurs, different region events need to be detected timely and valuable event evolution graph is needed, MLEM will works well. But there are some limitations of MLEM. Firstly, the event detection process is lack of domain knowledge, so MLEM will not work well in specific domain event detection task. Secondly, when calculating similarity between events, the graph structural characteristics are not fully utilized.

The future work is to integrate domain knowledge and advanced language representation model [47] into our model to obtain richer information about events and explore more event evolution patterns. Predicting the event trend based on evolution graph is also considered as a future work.

6 Acknowledgments

This work is supported by NSFC program (No. 61872022, 61421003, U1636123), SKLSDE-2018ZX-16 and partly by the Beijing Advanced Innovation Center for Big Data and Brain Computing.

References

1. Michael Mathioudakis and Nick Koudas. Twittermonitor:trend detection over the twitter stream. In *ACM SIGMOD International Conference on Management of Data*, pages 1155–1158, 2010.
2. Adrien Guille and CÃľcile Favre. Event detection, tracking, and visualization in twitter: a mention-anomaly-based approach. *Social Network Analysis & Mining*, 5(1):18, 2015.
3. Wei Xie, Feida Zhu, Jing Jiang, Ee Peng Lim, and Ke Wang. Topicsketch: Real-time bursty topic detection from twitter. *IEEE Transactions on Knowledge & Data Engineering*, 28(8):2216–2229, 2016.
4. Jianxin Li, Jianfeng Wen, Zhenying Tai, Richong Zhang, and Weiren Yu. Bursty event detection from microblog: a distributed and incremental approach. *Concurrency & Computation Practice & Experience*, 28(11):3115–3130, 2016.
5. Xiaoming Zhang, Xiaoming Chen, Yan Chen, Senzhang Wang, Zhoujun Li, and Jiali Xia. Event detection and popularity prediction in microblogging. *Neurocomputing*, 149:1469–1480, 2015.
6. Weiren Yu, Jianxin Li, Md Zakirul Alam Bhuiyan, Richong Zhang, and Jinpeng Huai. Ring: Real-time emerging anomaly monitoring system over text streams. *IEEE Transactions on Big Data*, 2017.
7. MÃąrio Cordeiro. Twitter event detection: combining wavelet analysis and topic inference summarization. In *DSIEâĂŹ12, the Doctoral Symposium on Informatics Engineering*, 2012.
8. Andreas Weiler, Michael Grossniklaus, and Marc H. Scholl. Event identification and tracking in social media streaming data. In *The Workshop on Multimodal Social Data Management*, pages 798–807, 2014.
9. Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. pages 1445–1456, 2013.
10. Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge & Data Engineering*, 26(12):2928–2941, 2014.
11. Peng Hao, Li Jianxin, Song Yangqiu, and Yang Qiang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *ACM World Wide Web*, 2018.
12. Edward Benson, Aria Haghighi, and Regina Barzilay. Event discovery in social media feeds. In *The Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, Usa*, pages 389–398, 2011.
13. Albert Angel, Nikos Sarkas, Nick Koudas, and Divesh Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story indentification. *Vldb Journal*, 23(2):175–199, 2012.
14. Manoj K. Agarwal, Manish Bhide, and Manish Bhide. *Real time discovery of dense clusters in highly dynamic graphs: identifying real world events in highly dynamic environments*. VLDB Endowment, 2012.
15. Hongyun Cai, Zi Huang, Divesh Srivastava, and Qing Zhang. Indexing evolving events from tweet streams. In *IEEE International Conference on Data Engineering*, pages 1538–1539, 2016.
16. Jingjing Wang, Wenzhu Tong, Hongkun Yu, Min Li, Xiuli Ma, Haoyan Cai, Tim Hanratty, and Jiawei Han. Mining multi-aspect reflection of news events in twitter: Discovery, linking and presentation. In *IEEE International Conference on Data Mining*, pages 429–438, 2016.
17. Francesco Bonchi, Ilaria Bordino, Francesco Gullo, and Giovanni Stilo. Identifying buzzing stories via anomalous temporal subgraph discovery. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 161–168, 2017.
18. Da Li, Srimat Chakradhar, and Michela Becchi. Grapid: A compilation and runtime framework for rapid prototyping of graph applications on many-core processors. In *IEEE International Conference on Parallel and Distributed Systems*, pages 174–182, 2015.
19. Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In *IEEE International Conferences on Big Data and Cloud*

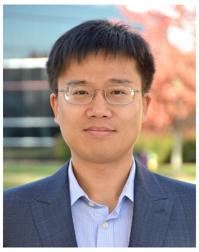
Computing, pages 477–484, 2016.

20. Da Li, Hancheng Wu, and Michela Becchi. Exploiting dynamic parallelism to efficiently support irregular nested loops on gpus. In *International Workshop on Code Optimisation for Multi and Many Cores*, pages 1–1, 2015.
21. Huan Truong, Da Li, Michela Becchi, Gavin Conant, and Kittisak Sajjapongse. A distributed cpu-gpu framework for pairwise alignments on large-scale sequence datasets. pages 329–338, 2013.
22. Da Li and Michela Becchi. Deploying graph algorithms on gpus: An adaptive solution. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 1013–1024, 2013.
23. Senzhang Wang, Xia Hu, Philip S. Yu, and Zhoujun Li. Mmrate:inferring multi-aspect diffusion networks with multi-pattern cascades. In *KDD*, pages 1246–1255, 2014.
24. Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July*, pages 497–506, 2009.
25. Christopher C. Yang, Xiaodong Shi, and Chih Ping Wei. Discovering event evolution graphs from news corpora. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(4):850–863, 2009.
26. Lee Pei, L. V. S Lakshmanan, and E. E Milios. Incremental cluster evolution tracking from highly dynamic network data. In *IEEE International Conference on Data Engineering*, pages 3–14, 2014.
27. Zhongyu Lu, Weiren Yu, Richong Zhang, Jianxin Li, and Hua Wei. Discovering event evolution chain in microblog. In *IEEE International Conference on High PERFORMANCE Computing and Communications*, pages 635–640, 2015.
28. Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Sigchi Conference on Human Factors in Computing Systems*, pages 227–236, 2011.
29. Pei Lee, Laks V. S Lakshmanan, and Evangelos E Milios. Event evolution tracking from streaming social posts. *Computer Science*, 2013.
30. Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3267–3273, 2017.
31. Hao Peng, Mengjiao Bao, Jianxin Li, Md Zakirulalam Bhuiyan, Yaopeng Liu, Yu He, and Erica Yang. Incremental term representation learning for social network analysis. *Future Generation Computer Systems*, 2017.
32. Duc T. Nguyen and Jai E. Jung. Real-time event detection for online behavioral analysis of big social data. *Future Generation Computer Systems*, 66:137–145, 2017.
33. Liu Yaopeng, Peng Hao, Guo Jie, He Tao, Li Xiong, Song Yangqiu, and Li. Jianxin. Event detection and evolution based on knowledge base. In *First Workshop on Knowledge Base Construction, Reasoning and Mining*, pages 38–39, 2018.
34. G Lejeune, R Brixtel, A Doucet, and N Lucas. Multilingual event extraction for epidemic detection. *Artificial Intelligence in Medicine*, 65(2):131–43, 2015.
35. Rodrigo Agerri, Itziar Aldabe, Egoitz Laparra, German Rigau, Antske Fokkens, Paul Huijgen, Marieke Van Erp, Ruben Izquierdo Bevia, Piek Vossen, and Anne Lyse Minard. Multilingual event detection using the newsreader pipelines. In *Cross-Platform Text Mining and Natural Language Processing Interoperability Workshop Co-Located with the Edition of the Language Resources and Evaluation Conference*, 2016.
36. Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. Generating event storylines from microblogs. In *ACM International Conference on Information and Knowledge Management*, pages 175–184, 2012.
37. Tao Ge, Wenzhe Pei, Heng Ji, Sujian Li, Baobao Chang, and Zhifang Sui. Bring you to the past: Automatic generation of topically relevant event chronicles. In *ACL*, pages 575–585, 2015.
38. Pengpeng Zhou, Bin Wu, and Zhen Cao. Emmbtt: A novel event evolution model based on tfxief and tdc in tracking news streams. In *IEEE Second International Conference on Data Science in Cyberspace*, pages 102–107, 2017.
39. Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David Mcclosky. The stanford corenlp natural language processing toolkit. In *Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014.
40. Weiren Yu, Charu C. Aggarwal, Shuai Ma, and Haixun Wang. On anomalous hotspot discovery in graph streams. In *IEEE International Conference on Data Mining*, pages 1271–1276, 2014.
41. Fergal Reid, Aaron Mcdaid, and Neil Hurley. Percolation computation in complex networks. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 274–281, 2012.
42. Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. In *Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A Meeting of Sigdat, A Special Interest Group of the Acl, Held in Conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411, 2004.
43. Jichang Zhao, Li Dong, Junjie Wu, and Ke Xu. Moodlens: an emoticon-based sentiment analysis system for chinese tweets. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1528–1531, 2012.
44. Sergey Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *IEEE International Conference on Data Mining*, pages 246–255, 2011.
45. Gergely Palla, Imre DerÃľnyi, IllÃľs Farkas, and TamÃąs Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814, 2005.
46. Ramesh Nallapati, Ao Feng, Fuchun Peng, and James Allan. Event threading within news topics. In *Thirteenth ACM International Conference on Information and Knowledge Management*, pages 446–453, 2004.
47. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yaopeng Liu, is currently a Ms.D. candidate at the Beijing Advanced Innovation Center for Big Data and Brain Computing and State Key Laboratory of Software Development Environment in Beihang University(BUAA), Beijing, China. His research interests include representation learning and data mining. E-mail: liuyp@act.buaa.edu.cn.

Hao Peng, is currently a Ph.D. candidate at the Beijing Advanced Innovation Center for Big Data and Brain Computing, and State Key Laboratory of Software Development Environment in Beihang University(BUAA), Beijing, China. His research interests include representation learning, social network analysis and text mining. E-mail: penghao@act.buaa.edu.cn.

Jianxin Li, is a professor at the Beijing Advanced Innovation Center for Big Data and Brain Computing, and the State Key Laboratory of Software Development Environment in Beihang University(BUAA), Beijing, China. His current research interests include big data, distributed system, virtualization, trustworthy computing and network security. E-mail: lijx@act.buaa.edu.cn. The corresponding author.



Yangqiu Song, is an assistant professor at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. His research interest is using machine learning and data mining techniques to extract and infer insightful knowledge from big data. E-mail: yqsong@act.buaa.edu.cn.

Xiong Li, received the PhD degree in pattern recognition and intelligence system from Shanghai Jiao Tong University, China, in 2013. He is currently a senior engineer in National Computer Network Emergency Response Technical Team, China. His research interests include hybrid generative discriminative learning and probabilistic graphical model.