Event Detection and Evolution Based on Knowledge Base

Yaopeng Liu* Hao Peng* Jie Guo* Tao He* Xiong Li†
Yangqiu Song‡ Jianxin Li*
*School of Computer Science & Engineering, Beihang University, Beijing, China
†National Computer Network Emergency Response Technical Team/Coordination Center of China
‡Department of Computer Science & Engineering, Hong Kong University of Science and Technology, Hong Kong
{liuyp,penghao,guojie,hetao,lijx}@act.buaa.edu.cn li.xiong@foxmail.com yqsong@cse.ust.hk

ABSTRACT

Social media and News media have become an important platform where people obtain latest events. Much research has been done to detect topics and events in social or news streams, but lack of comprehensiveness. In this paper, we propose a Knowledge-based Event Mining (KEM) model to automatically detect events and generate evolution chain in multi-channel text streams. Specifically, we merge the same entities and similar phrases by knowledge base and incremental word2vec model. We adopt a 7-tuple event description model to display events comprehensively and analyze their relationships. Finally, we generate an evolution chain for each event incrementally. Our evaluation on a massive human-generated dataset containing real world events demonstrates that our new model KEM outperforms the baseline method both in efficiency and effectiveness.

1 INTRODUCTION

Real-world occurrences reported in internet are also called events. Events detected from official and social media hold critical messages that describe the situations during real-world occurrences. Mining such knowledge is attractive to both policy makers monitoring public sentiment and ordinary people obtaining event details. Consequently, Event Detection and Evolution have received much attention in recent years.

Previous event detection techniques include Bursty Detection [10, 18, 23, 24, 31, 37], Topic Model [3, 7, 8, 11, 30, 35] and other Clustering Algorithms [1, 2, 4–6, 14–17, 19, 33, 34, 36, 41]. Merging duplicates can be challenging in this task because different phrases may share the same meaning and phrase semantics may change over time. For example, *"American president"* was the most relavant to *"Obama"* in 2010s, but related to *"Trump"* recently.

Moreover, all of these methods are single-channel-oriented, lack of real-time and comprehensiveness. Because in some cases, events are perceived on social media earlier than traditional news media, but sometimes quite the reverse. For example, disastrous events like Las Vegas Shooting was detected promptly in tweets. On the contrary, political or military affairs are noticed only until the news appear like the executive order. In order to ensure the real-time capability of the event detection model, we collect the multi-channel text streams as the input of the model. Information redundancy in long-text and insufficiency in short-text making hybrid-length text streams processing the second challenge.

Limited research has been conducted on event evolution due to its diversity. Previous studies represent event evolution by graph [13, 27, 38], but are not fit for short-text or long spanning event evolution. Evolution on short-text streams has attracted much attention in recent years, some represent event evolution by volume on time dimension [6, 12, 22], but considerably important events are discontinuous in time. Some stuidies discover an event chain [20, 35], but cannot show the evolution patterns.

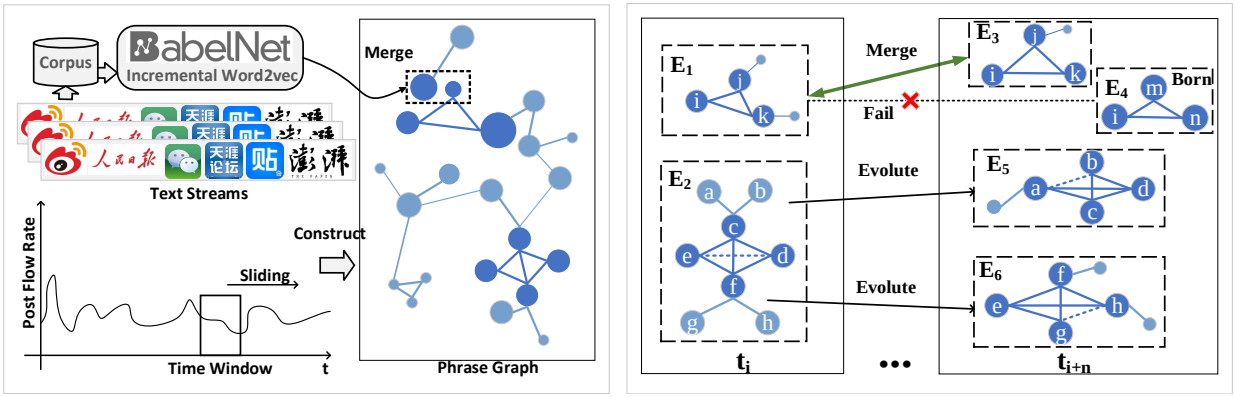
Influential events emerge and evolve in multi-channels, such as Weibo, News websites and Forums. Official and social media describe events from different perspectives. All of the above provides a complementary view of an event: Reasonably objective reports and full view of opinions from home and abroad. Collecting them together to provide a complete picture of an event can be a crucial issue for both policy makers and ordinary people. In addition, a "panoramic view" of the event will help users know how did it get to here and make instant decision wisely. In sum, our major challenges are merging multi-channel sources, hybrid-length text processing, semantic drift problem and evolution chain generation.

In order to handle these challenges, we propose a Knowledge-based Event Mining (KEM) model to automatically detect events and incrementally generate evolution chain. We first model text streams as an evolving phrase graph, and focus on significant phrases and cliques they form. Then we introduce BabelNet [25] and incremental word2vec model into event detection model to merge duplicates at semantic level and solve semantic drift problem. We employ Incrementally learning of the Hierarchical Softmax Function [29] instead of Matrix Factorization [28] to save space complexity. BabelNet is a very large, wide-coverage multilingual knowledge base. Secondly, we present a novel 7-tuple model based on event attributes, including time, location, participants, keywords, emotion, summary and most-related posts. Posts are microblogs, breaking news and forum messages. Finally, we propose a method to generate event evolution chain automatically based on multiple similarity measures. We utilize BabelNet and word2vec to divide events into subgraphs.

We show an overview of the major steps of our KEM model for event detection and evolution from hybrid-length text streams in Figure 1. Note that as time rolls on, the phrase graph will be updated

KBCOM2018, Los Angeles, California

DOI: 10.475/123_4



(a) Event Detection (b) Event Evolution

Figure 1: (a) Phrase graph captures strong correlation between phrases. Node size represents the phrase frequency. Edge thickness indicates the correlation strength. Each event is annotated by a phrase subgraph, which is dark colored. (b) As time rolls on, typical evolution patterns include born, evolute and merge. Each phrase graph is extended with detail phrases, which are light colored.

incrementally at each moment. Phrase graphs in the detection process and the evolution process share the same key nodes, which are made fully connected in the latter process.

The rest of this paper is organized as follows. Section 2 provides a review of the related work. Section 3 presents our model and algorithms. Section 4 evaluates the proposed approach and compares it with the existing methods. Finally, Section 5 concludes the work.

2 RELATED WORK

In this section, we briefly introduce the related work published in the areas of event detection and event evolution, which can be divided into one of the following categories.

2.1 Event Detection in Social Streams

Impressive efforts have been devoted to the detection of significant events in social streams. Mathioudakis et al. [23] introduced a monitor that identifies events by sharp increase of keywords frequency in a specified time slice. However, this monitor cannot distinguish different events sharing the same keywords in bursty flow. Angel et al. [2] and Agarwal et al. [1] both described the social streams using a highly dynamic entity graph, from which they extracted dense subgraphs as events. These methods suffer from the loss of single-entity events or only post attributes like action of the entities. Cheng et al. [7] modeled the word co-occurrence patterns in the corpus to learn topics, making emerging topics inference effectively. Topic models have also been refined to apply to short texts, but still require prior knowledge, the topics are limited and not suitable for our problems. Our KEM model employs BabelNet and incremental word2vec model to introduce semantic information into phrase graph generation and discover more fine-grained events.

2.2 Event Tracking in Social Streams

In general, Event Tracking is implemented online. Ge et al. [9] presented a learning-to-rank model to generate a topically relevant event chronicles for certain period. But this method requires predefined topics, which makes it not applicable to arbitrary event tracking, since future events may belong to unknown topics. Pei et al. [27] proposed a method to build story-teller for streaming social content, it constructs a sketch graph to summarize information in the dynamic network by monitoring through a fading time window. It recognizes evolution patterns by monitoring the development of subgraphs. This method can only track event evolution in adjacent windows, so it's not suitable for events spanning over a long time.

2.3 Event Evolution with Correlation Building Methods

Yang et al. [38] defined an event evolution scoring function to discover evolution relationships. This function utilizes the timestamp, content similarity, temporal proximity, and document distributional proximity to model the event evolution relationships, while timestamp can be misleading for long-term spanning events and it doesn't take vital attributes into consideration. Weiler et al. [35] used word matrix, Lu et al. [20] took location and participants into consideration and Zhou et al. [40] utilized TFIEF and Temporal Distance Cost factor to measure the event relationships and generate an evolution chain. These methods miss semantic information and evolution patterns. Our method belongs to this category and we design a new events correlation building method in this paper to generate an evolution chain.

In our approach, we adopt metrics including phrase subgraphs, location and participants to evaluate event relationships. The key issue of recognizing relationships is phrase subgraphs similarity measurement. We introduce incremental word2vec model to measure it and multilingual knowledge base to discover potential relationships.

3 MODEL AND ALGORITHMS

In this section, we introduce our method for event detection and event evolution chain generation from multi-channel text streams based on Knowledge Base, which we call KEM model.

3.1 Preliminaries

Phrase Graph Definition. Given text streams in a specified time slice, we define the phrase graph as $G(\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ represents phrases extracted from the text streams and we accept multiple entities or verbs sharing the same meaning on one node, $\mathbb{E}$ represents edges corresponding to co-occurrence relationships between phrases in a text sliding window.

Weight decay Formulation. Given a text sliding window, we define the decay of edge weight W as:

$$W(d) = W(0) \cdot 2^{-\lambda \cdot \frac{d}{l}} \tag{1}$$

where $W(0)$ is a weight constant, λ is the decay factor, d is the phrase count text window slide away from the beginning, l is the width of window.

Word Semantic Similarity. Given two words $w1$ and $w2$, we define the semantic similarity between them as cosine distance:

$$Sim(w1, w2) = v_{w1} \cdot v_{w2} \tag{2}$$

where $v(w)$ is the unit vector of word w in word2vec model.

Event Definition. We denote an event as an 7-tuple:

$$E =< t; desc; loc; par; key; emo; posts > \tag{3}$$

where t is the timestamp when the event is detected. $desc$ is the summary of the event, which is usually a short sentence. loc is the location where the event happened. par is a set of participants mainly involved in the event. key is a set of key phrases of the event. emo is the emotion revealed by the event, which can be positive, negative or neutral. $posts$ is a set of posts related to the event, with limit size of 5 after duplication removal.

Evolution Chain Definition. Given a latest event E_0, we donate the evolution chain of E_0 as C_0, represented as a sequential pattern:

$$E_n \rightarrow E_{n-1} \leftrightarrow E_{n-2} \rightarrow \cdots \rightarrow E_1 \rightarrow E_0 \tag{4}$$

where $E_i \rightarrow E_j$ means E_j is evoluted from E_i, $E_i \leftrightarrow E_j$ means E_j is merged by E_i. Chain C_0 traces back the whole development of event E_0 along the timeline.

3.2 Phrase Graph Generation

Text streams within the same time sliding window construct a phrase graph shown in Figure 1(a). Taking into account the speed of propagation in social media and news channel, we set this interval to 600 seconds. There are a lot of noises from multi-channel text streams, such as Twitter or Weibo, because they are usually written in an informal way. Therefore, we perform standard text processing tasks such as tokenization, stopword, punctuation, special character removal, stemming and Name Entity Recognition using the Stanford CoreNLP tool [21]. Specifically, entities and verbs are retained and texts less than three phrases are eliminated to reduce noises.

Then we merge the same entities in different manifestations like *North Korean* and *DPRK*, *Russian President* and *Putin*. We employ BabelNet to merge multiple entities into one node and represent it with a unique ID from the knowledge base. Therefore, an ID sequence is generated for word2vec training and edge connections.

Useful information is mainly concentrated in the front part of the long-text posts, far distant phrases may be irrelevant. For example, some people just put multiple hashtages corresponding to real-time hot events into one microblog to get on the top search list, in order to promote the product or just comment on them together.

Under this condition, simply drawing edges between phrases by co-occurrence relation in a post will produce interfering information, so we introduce a text sliding window to draw weighted edges. If two phrases co-occur in the same text window, a weighted edge is drawn between the two associated nodes following Eq (1). For multiple co-occurrence in a single post, we update the weight for this post by accumulation.

We introduce knowledge base to discover strong relationship outside the sliding window such as *is a*, *child*, *spouse*, *founder*. Then we detect strong relationship between two phrases, if there is an edge for this post between them, we update the weight to $W(0)$, otherwise we draw an edge with weight $W(0)$.

After going through all posts, if there are edges with same endpoints distinguished by post ID, we merge them by adding their weights together. Consequently, phrases with the same meaning constitute $\mathbb{V}$ and phrase co-occurrence or strong correlation construct $\mathbb{E}$ defined in 3.1.

3.3 Event Detection Model

The Event Detection part of our KEM model detect anomalous hot phrases on the phrase graph using the method proposed by Yu et al. [39]. In particular, we keep hot phrases and limited edges with top weights extended from them, to construct a small anomalous subgraph. Finally, we adopt an optimized overlapping community finding algorithm [32] on the subgraph to find out events from these hot phrases. This method defines community as k-clique: a set of fully-connected k nodes. Cliques that share $k-1$ nodes will be merged. We regard each community as an event and represent it with key phrases in the community temporarily.

Furthermore, we draw edges to merge cliques telling the same thing using word2vec, a useful model to calculate the cosine distance of words and represent them. We cannot simply adopt word2vec model to merge nodes, because the cosine distance between phrase vectors represents the possibility that sentences remain same meaning after substitution, and it is not accurate enough at the semantic level. Since the co-occurrence possibility changes from time to time, we train hierarchical softmax function after every 12 time slices on old corpus and new corpus generated in 3.2 incrementally.

When going through the anomalous subgraph, if cosine distance between two nodes exceeds threshold ϕ, we will link edges from each node to nodes connected with the other. We explain the drawing process in Figure 2. If $cos(b, e)$ exceeds ϕ, we connect b, d, b, f and e, c, merging cliques abc, ade and node f to $abcdef$. For example, nodes b and e could be *kill* and *murder*, *injure* and *wound*.

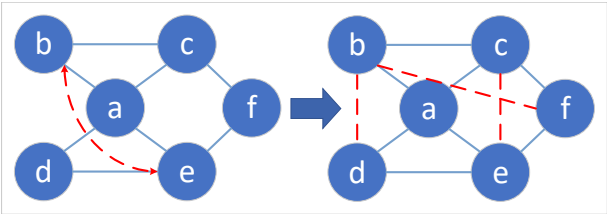


Figure 2: The solid line represents co-occurrence or strong relationship, the dotted bidirectional arrow means cosine distance of two nodes exceeds ϕ, the dotted line represents the edge connected through word2vec model.

After finishing above steps, we fill the 7-tuple to make events easy to understand. The key phrases are naturally set as key. We get t, $desc$ and $posts$ using the method proposed in [20]. In particular, if multiple news exist in the posts, the priority is to set $desc$ as the news headline containing most of the key phrases. Then we query the knowledge base to discover loc and par from the key phrases. The most frequently mentioned location is set as loc, people and

organizations are put into set par. If the search from key phrases fails, we will go through related posts and fill loc and par again. Finally, we employ Bayes model for sentiment classification, the details of which is beyond the scope of this paper.

Our event description model describes events from 7 aspects, which is informative and understandable. An example is illustrated in Table 1, which is about South Korea missile launch in Sept.15.2017.

Table 1: An Example of Event Description Model

t	2017/09/15 09:10
$desc$	South Korean army launched a ballistic missile against North Korea.
loc	South Korea
par	Kim Jong-un
key	South Korea, ballistic missiles, peninsula, North Korea
emo	negative
$posts$	According to the North Korea Joint Staff, North Korea launched a missile to the eastern waters of the peninsula this morning, the South Korean army immediately launched the $\cdots$

3.4 Event Correlation Building Method

To build the event evolution graph, the main challenge is how to evaluate the relationship between two events. Given two events E_i and E_j, which are separately denoted as $< t_i, desc_i, loc_i, par_i, key_i, emo_i, posts_i >$ and $< t_j, desc_j, loc_j, par_j, key_j, emo_j, posts_j >$. We define event similarity score function as:

$$\begin{aligned} Sim(E_i, E_j) = \alpha \cdot Sim_{subgraphs}(key_i, posts_i, key_j, posts_j) \\ + \beta \cdot Sim_{loc}(loc_i, loc_j) + \gamma \cdot Sim_{par}(par_i, par_j) \end{aligned} \tag{5}$$

where $Sim_{subgraphs}$, Sim_{loc}, Sim_{par} denote similarity measures of phrase subgraphs, location and participants respectively. α, β, γ are weight coefficients of these measures subjected to $\alpha + \beta + \gamma = 1$.

$Sim_{subgraphs}$ evaluates the similarity between events' phrase subgraphs. We consider that global text similarity of events is not a good choice and take $Sim_{subgraphs}$ as a significant feature, because an event may has numerous sides and only part of them are useful. We first construct sketch graph using key phrases and detail phrases. Key phrases naturally inherit from phrases in the community during the detection process. Detail phrases are top 10 phrases in related posts co-occur with key phrases in a text sliding window. Then we merge phrases with the same meaning through the method in 3.2. We make key phrases fully connected and draw edges with each phrase in a text sliding window. The final sketch graph is shown in Figure 1(b), the dotted edges and light colored edges between key nodes are added in this step.

Furthermore, we regard each three connected nodes containing at least one key node as a subgraph, and calculate similarity of each subgraph pair Sim_T. Each phrase could be represented by a unit vector using word2vec, and Sim_T is calculated by average cosine distance of phrase on each subgraph. We represent each triangle as a triple vector $T_i =< v_{i,1}, v_{i,2}, v_{i,3} >$, where $v_{i,j}$ denotes the vector of phrase j in T_i, and Sim_T is defined as:

$$Sim_T(T_i, T_j) = avg(v_{i,1}, v_{i,2}, v_{i,3}) \cdot avg(v_{j,1}, v_{j,2}, v_{j,3}) \tag{6}$$

On the basis of above equations, $Sim_{subgraphs}$ is defined as the maximum value among all $Sim_T(T_i, T_j)$ in two events:

$$Sim_{subgraphs} = \max_{i \in 1,n, j \in 1,m} Sim_T(T_i, T_j) \tag{7}$$

where n and m are the count of triangles in two events.

Events with evolution relationships mostly happen in the same place. Sim_{loc} is employed to evaluate the similarity of events location based on this assumption:

$$Sim_{loc}(loc_i, loc_j) = \begin{cases} 1 & if \ \ loc_i \ \ equals \ \ loc_j; \\ 0 & otherwise. \end{cases} \tag{8}$$

Similarly, we define Sim_{par} as the Jaccard Coefficient of participant sets:

$$Sim_(par)(par_i, par_j) = \frac{par_i \cap par_j}{par_i \cup par_j} \tag{9}$$

After the similarity of two events was calculated, we define that E_j is evolved from E_i if $Sim(E_i, E_j) > \epsilon$ and $t_i < t_j$, i.e.,

$$E_i \rightarrow E_j \ \ if \begin{cases} Sim(E_i, E_j) > \epsilon \\ t_i < t_j \end{cases} \tag{10}$$

We tune ϵ to 0.45 in this paper through experiments on a manually annotated dataset.

3.5 Event Evolution Chain Generation

In this chapter, we focus on building the whole evolution chain for a given event E_0. Since events may evolute into a very different event, once given a new event, we find each event that has evolution relationship with it and patch these events to the chain which the most-related event falls in. It makes the evolution chain expand incrementally and saves plenty of time.

First, we get events set by searching event index and adopt two-layer filtering method introduced in [20] to reduce computational complexity. After filtering, we get a candidate events set CS with maximum size 20 for input event E_0. Then we head to find all events E' for input event E_0 meeting the condition: $E' \rightarrow E_0$, making a set of events: S_E. Afterwards, we get evolution chain of event E_{max} with the maximum similarity in S_E, remove events from S_E that already occur in the chain, patch all events in S_E to it, and reorder S_E in time order. Finally, we discover evolute and merge relationships between events.

We define merging relationship as same events detected in continuous time span. We adopt thresholds ϵ' set as 0.8 to merge same events. We merge events back to the earliest one and represent its t as a time span striding across multi time slice, because breaking news or buzz topic can keep high heat through a long period of time.

Algorithm 1 illustrates the method to generate event evolution chain for given event E.

4 EXPERIMENTS

In this section, we first describe the dataset that we used for our experiments and the parameters we adopt. We test our model KEM from both efficiency and effectiveness against baseline methods.

Dataset: We have been collecting multi-channel text data and detecting events since Feb.12.2016. Raw data from multi sources including Weibo, Wechat, Forum and News Media is stored in HBase and indexed by Elasticsearch. Representative media is shown in Figure 1(a). Until now there has been about 2.9 billion weibos, 4.1

Algorithm 1 Event Evolution Chain Generation.

Input: E
Output: Array $Chain[]$

$sim_{max} = 0$;
$pos_{max} = 0$;
Get events set CS by searching event index and two-layer filtering;
for $i = 1; i < size(CS); i++$ **do**
  Calculate $Sim(E, CS[i])$;
  if $Sim(E, CS[i]) > \epsilon$ **then**
    Insert $CS[i]$ to the S_E;
    if $sim_{max} < Sim(E, CS[i])$ **then**
      $sim_{max} = Sim(E, CS[i])$;
      $pos_{max} = size(S_E) - 1$;
Get $Chain[]$ of $S_E[pos_{max}]$ by querying event index;
Remove duplicates and put events from S_E to $Chain[]$;
Sort $Chain[]$ by time ascending order;
$j = 1$;
repeat
  Calculate $Sim(Chain[j-1], Chain[j])$;
  if $Sim(Chain[j-1], Chain[j]) > \epsilon'$ **then**
    Merge $Chain[j]$ to $Chain[j-1]$;
    Remove $Chain[j]$ from $Chain$;
  else
    $j++$;
until $j > size(S_E)$
return $Chain[]$;

million news, 6.7 million forum messages collected by crawlers through API or web page, from which about 2 million events have been detected .

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

Table 2: Parameters Setting

Parameter	Default	Description
$W(0)$	1	coefficient of edge weight
λ	0.05	decay factor of edge weight
l	10	width of the text sliding window
α	0.55	coefficient of subgraphs similarity
β	0.1	coefficient of location similarity
γ	0.15	coefficient of participant similarity
ϵ	0.45	threshold of evolution relationship
ϵ'	0.8	threshold for merging events
ϕ	0.78	threshold of merging phrases
n_hashes	5	number of hashes in LSH
n_tables	5	number of hash tables in LSH
$n\_neighbours$	10	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 posts in one day are randomly picked out from our post database and treated as the input of Event Detection Model.

Baseline. Works like [20] first detect trending keywords using [39] and then adopt overlapping community detection method [26] to discover events from these trending keywords in a time slice.

Efficiency

In this part we evaluate the efficiency of our model against [20]. Figure 3 shows the distribution of event count in each time slice by our method and baseline method. Figure 4 shows the distribution of average detection time for each detected event by the two methods.

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 56.6s with baseline method and 85.3s with our method. Meanwhile, the average count of events we detect is 25.7, it is about 1.95 times as mush as baseline method, the corresponding average detection time for each event is 3.31s and 3.99s on each method. In other words, the detection model increases the event count by 81.35% and improves the computing speed by 13.94%. This is mainly because baseline method connects all words in a single long report, and that makes the word graph much complicated. Furthermore, the clique baseline method detected contains too many words, it may fail retrieving the posts with too many interference, so lots of computation time is wasted.

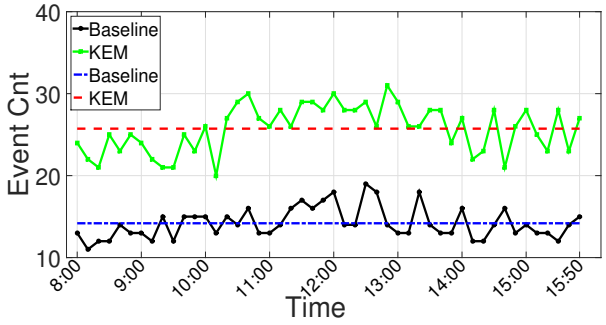


Figure 3: Distribution of Detected Event Quantity.

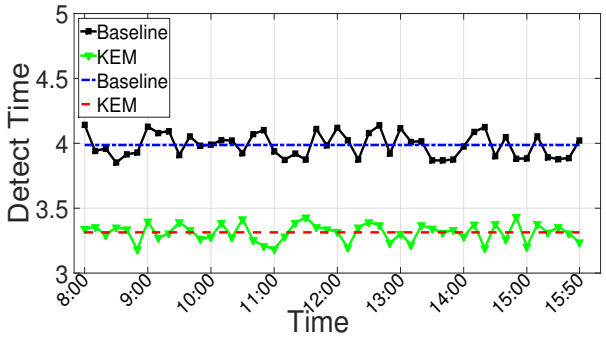


Figure 4: Distribution of Detection Time.

Effectiveness

In this part we compare our method and baseline method in terms of detection effectiveness.

Ground Truth. We extract 300 events from authoritative news articles in September, 2017. These news all have enough related posts obviously and news title is treated as summary for each event.

Evaluation. We use the standard metrics **Precision(P)**, **Recall(R)**, **F1-Measure(F1)** and average time delay $\Delta\mathbf{T}$ to quantize the effectiveness of our model. They are calculated as follows:

$$\begin{cases} P = \dfrac{|G \cap C|}{|C|} \\ R = \dfrac{|G \cap C|}{|G|} \\ F1 = \dfrac{|2 \cdot PR|}{|P + R|} \\ \Delta T = \dfrac{\sum_{E_i \in G \cap C}(T_{i,detect} - T_{i,emerge})}{|G \cap C|} \end{cases} \tag{11}$$

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 the two methods over the dataset. The difference of recalls is much larger than precisions mainly because the merging of same entities in 3.2 and edge reconnection in 3.3. These two operations lift heat of each node on the graph to an anomalous level so more events are detected and the accuracy is improved also. The difference between time delay is because baseline method is unable to process multi-channel sources and it takes time for event to spread across channels. Moreover, the weakness for processing long text lead to the gap of effectiveness. It is clear that our method achieve superior effectiveness over baseline method.

Table 3: Detection Effectiveness Result

Method	P	R	F1	ΔT/min
Baseline	0.5833	0.3733	0.4553	36.46
KEM	0.6133	0.6767	0.6434	14.23

4.2 Event Evolution

In this section we evaluate the effectiveness of our event evolution model. We compare our algorithm with a baseline method and ground truth generated from news to evaluate the effectiveness.

Ground Truth. 300 events are picked out from our event database and treated as the input of the evolution model. 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 event and manually put them on an evolution chain follow the time order.

Baseline. Works like [20] also use the combination of multiple features to measure the relationships between two events. Content similarity and temporal proximity are most widely used as a classic method and the state-of-art method combine similarity of location, participants and posts together, the latter beats the former evidently in effectiveness aspect. Therefore we adopt the latter method as the baseline. The similarity between events is computed as follows:

$$\begin{aligned} Sim(E_i, E_j) &= \alpha \cdot Sim_{posts}(posts_i, posts_j) \\ &+ \beta \cdot Sim_{loc}(loc_i, loc_j) + \gamma \cdot Sim_{par}(par_i, par_j) \end{aligned} \tag{12}$$

where $Sim_{posts}(posts_i, posts_j)$ is related posts' similarity between E_i and E_j and it's given by:

$$Sim_{posts}(posts_i, posts_j) = \frac{\sum_{p_k \in posts_i} \sum_{p_l \in posts_j} cos\_sim(p_k, p_l)}{|posts_i| \cdot |posts_j|} \tag{13}$$

where $cos\_sim(p_i, p_j)$ is defined as follows:

$$Sim_{d_i, d_j} = \frac{\sum_{k=1}^{N} w_{i,k} \cdot w_{j,k}}{\sqrt{\sum_{k=1}^{N}(w_{i,k})^2} \cdot \sqrt{\sum_{k=1}^{N}(w_{j,k})^2}} \tag{14}$$

where $w_{i,j}$ donates the frequency of term j in p_i and N is the size of vocabulary.

Specially, [20] didn't extend the evolution chain incrementally and 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 (11).

Table 4 shows the **P, R, F1** of our methods with different settings of parameters α, β, γ and baseline method comparing to the ground truth. For our method, we find that the setting $\alpha = 0.6, \beta = 0.1, \gamma = 0.3$ has the highest recall, the setting $\alpha = 0.7, \beta = 0.1, \gamma = 0.2$, which is the default setting, has the highest precision and **F1** score, and outperforms the baseline on each metrics.

Table 4: Evolution Effectiveness Result

	α	β	γ	P	R	F1
KEM	1	0.0	0.0	0.5569	0.2915	0.3827
	0.9	0.1	0.0	0.4700	0.3197	0.3806
	0.9	0.0	0.1	0.5805	0.4295	0.4937
	0.8	0.1	0.1	0.5574	0.4730	0.5118
	0.7	0.1	0.2	**0.5896**	0.5489	**0.5685**
	0.7	0.2	0.1	0.5619	0.5649	0.5634
	0.6	0.2	0.2	0.5159	0.5834	0.5476
	0.6	0.1	0.3	0.4855	**0.5947**	0.5346
	0.6	0.3	0.1	0.4731	0.5646	0.5148
Baseline	0.7	0.1	0.2	0.5441	0.4451	0.4897

Anomalously, we find the recall for baseline method in our experiment is much lower than our method, we check the ground truth data and find out that if two events share similar subgraphs which means the the core elements of the event are similar, we add an evolution relationship between them. But the baseline method consider them as irrelevant events because it doesn't reach the evolution threshold. In our method, events don't need to be similar enough to be in one chain, which leads to the increase of recall.

5 CONCLUSION

In this paper, we propose a novel model called KEM for knowledge-based event detection and evolution chain generation. We utilize knowledge base and incremental word2vec to merge duplicates in detection process. We combine the similarity measures of phrase subgraphs, location and participants to evaluate the relationships among events using knowledge base. We adopt two-layer filtering to get the candidate events for each given event to save calculation

time. Experiments show the high performance of our KEM model in efficiency and effectiveness. For efficiency, our detection model improves computing speed by 13.94% for each event. For effectiveness, our KEM model outperforms baseline on both precision and recall.

The future work is to visualize events as evolution graph to aid human perception and explore more event evolution patterns.

6 ACKNOWLEDGMENTS

This work is supported by China 973 Fundamental R&D Program (No.2014CB340300), NSFC program (No.61403090,61772151, U1636123) and Beijing Advanced Innovation Center for Big Data and Brain Computing.

REFERENCES

[1] Manoj K. Agarwal, Manish Bhide, and Manish Bhide. 2012. *Real time discovery of dense clusters in highly dynamic graphs: identifying real world events in highly dynamic environments*. VLDB Endowment. 980–991 pages.

[2] Albert Angel, Nikos Sarkas, Nick Koudas, and Divesh Srivastava. 2012. Dense subgraph maintenance under streaming edge weight updates for real-time story indentification. *Vldb Journal* 23, 2 (2012), 175–199.

[3] Hila Becker, Mor Naaman, and Luis Gravano. 2011. Beyond Trending Topics: Real-World Event Identification on Twitter.. In *International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July*.

[4] Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. 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*. 389–398.

[5] Francesco Bonchi, Ilaria Bordino, Francesco Gullo, and Giovanni Stilo. 2017. Identifying Buzzing Stories via Anomalous Temporal Subgraph Discovery. In *Ieee/wic/acm International Conference on Web Intelligence*. 161–168.

[6] Hongyun Cai, Zi Huang, Divesh Srivastava, and Qing Zhang. 2016. Indexing evolving events from tweet streams. In *IEEE International Conference on Data Engineering*. 1538–1539.

[7] Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. 2014. BTM: Topic Modeling over Short Texts. *IEEE Transactions on Knowledge & Data Engineering* 26, 12 (2014), 2928–2941.

[8] Mrio Cordeiro. 2012. Twitter event detection: combining wavelet analysis and topic inference summarization. In *DSIE12, the Doctoral Symposium on Informatics Engineering*.

[9] Tao Ge, Wenzhe Pei, Heng Ji, Sujian Li, Baobao Chang, and Zhifang Sui. 2015. Bring you to the past: Automatic Generation of Topically Relevant Event Chronicles. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*. 575–585.

[10] Adrien Guille and Ccile Favre. 2015. Event detection, tracking, and visualization in Twitter: a mention-anomaly-based approach. *Social Network Analysis & Mining* 5, 1 (2015), 18.

[11] Peng Hao, Li Jianxin, Song Yangqiu, and Yang Qiang. 2018. Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In *ACM World Wide Web*.

[12] Pei Lee, Laks V. S Lakshmanan, and Evangelos E Milios. 2013. Event Evolution Tracking from Streaming Social Posts. *Computer Science* (2013).

[13] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. 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*. 497–506.

[14] Da Li and Michela Becchi. 2013. Deploying Graph Algorithms on GPUs: An Adaptive Solution. In *IEEE International Symposium on Parallel and Distributed Processing*. 1013–1024.

[15] Da Li, Srimat Chakradhar, and Michela Becchi. 2015. 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*. 174–182.

[16] Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. 2016. Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs. In *IEEE International Conferences on Big Data and Cloud Computing*. 477–484.

[17] Da Li, Hancheng Wu, and Michela Becchi. 2015. Exploiting Dynamic Parallelism to Efficiently Support Irregular Nested Loops on GPUs. In *International Workshop on Code Optimisation for Multi and Many Cores*. 1–1.

[18] Jianxin Li, Jianfeng Wen, Zhenying Tai, Richong Zhang, and Weiren Yu. 2016. Bursty event detection from microblog: a distributed and incremental approach. *Concurrency & Computation Practice & Experience* 28, 11 (2016), 3115–3130.

[19] Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu. 2011. *Towards effective event detection, tracking and summarization on microblog data*. Springer Berlin Heidelberg. 652–663 pages.

[20] Zhongyu Lu, Weiren Yu, Richong Zhang, Jianxin Li, and Hua Wei. 2015. Discovering Event Evolution Chain in Microblog. In *IEEE International Conference on High PERFORMANCE Computing and Communications*. 635–640.

[21] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David Mcclosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Meeting of the Association for Computational Linguistics: System Demonstrations*.

[22] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. 2011. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Sigchi Conference on Human Factors in Computing Systems*. 227–236.

[23] Michael Mathioudakis and Nick Koudas. 2010. TwitterMonitor:trend detection over the twitter stream. In *ACM SIGMOD International Conference on Management of Data*. 1155–1158.

[24] Mor Naaman, Hila Becker, and Luis Gravano. 2011. Hip and trendy: Characterizing emerging trends on Twitter. *JASIST* 62, 5 (2011), 902–918. DOI: http://dx.doi.org/10.1002/asi.21489

[25] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193, 6 (2012), 217–250.

[26] Gergely Palla, Imre Dernyi, Ills Farkas, and Tams Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (2005), 814.

[27] Lee Pei, L. V. S Lakshmanan, and E. E Milios. 2014. Incremental cluster evolution tracking from highly dynamic network data. In *IEEE International Conference on Data Engineering*. 3–14.

[28] Hao Peng, Mengjiao Bao, Jianxin Li, Md Zakirulalam Bhuiyan, Yaopeng Liu, Yu He, and Erica Yang. 2017. Incremental term representation learning for social network analysis. *Future Generation Computer Systems* (2017).

[29] Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. 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*. 3267–3273. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14732

[30] Swit Phuvipadawat and Tsuyoshi Murata. 2010. Breaking News Detection and Tracking in Twitter. In *Ieee/wic/acm International Conference on Web Intelligence and Intelligent Agent Technology*. 120–123.

[31] Ana Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. 2011. Extracting events and event descriptions from Twitter. In *International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April*. 105–106.

[32] Fergal Reid, Aaron Mcdaid, and Neil Hurley. 2012. Percolation Computation in Complex Networks. In *Ieee/acm International Conference on Advances in Social Networks Analysis and Mining*. 274–281.

[33] Huan Truong, Da Li, Michela Becchi, Gavin Conant, and Kittisak Sajjapongse. 2013. A distributed CPU-GPU framework for pairwise alignments on large-scale sequence datasets. (2013), 329–338.

[34] Jingjing Wang, Wenzhu Tong, Hongkun Yu, Min Li, Xiuli Ma, Haoyan Cai, Tim Hanratty, and Jiawei Han. 2016. Mining Multi-Aspect Reflection of News Events in Twitter: Discovery, Linking and Presentation. In *IEEE International Conference on Data Mining*. 429–438.

[35] Andreas Weiler, Michael Grossniklaus, and Marc H. Scholl. 2014. Event Identification and Tracking in Social Media Streaming Data. In *The Workshop on Multimodal Social Data Management*. 798–807.

[36] Jianshu Weng and Bu Sung Lee. 2011. Event Detection in Twitter. In *International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July*. 311–312.

[37] Wei Xie, Feida Zhu, Jing Jiang, Ee Peng Lim, and Ke Wang. 2016. TopicSketch: Real-Time Bursty Topic Detection from Twitter. *IEEE Transactions on Knowledge & Data Engineering* 28, 8 (2016), 2216–2229.

[38] Christopher C. Yang, Xiaodong Shi, and Chih Ping Wei. 2009. Discovering Event Evolution Graphs From News Corpora. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 39, 4 (2009), 850–863.

[39] Weiren Yu, Charu C. Aggarwal, Shuai Ma, and Haixun Wang. 2014. On Anomalous Hotspot Discovery in Graph Streams. In *IEEE International Conference on Data Mining*. 1271–1276.

[40] Pengpeng Zhou, Bin Wu, and Zhen Cao. 2017. 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*. 102–107.

[41] Xiangmin Zhou and Lei Chen. 2013. Chen, L.: Event detection over twitter social media streams. VLDB J. 23(3), 381-400. *Vldb Journal* 23, 3 (2013), 381–400.