2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS)

Discovering Event Evolution Chain in Microblog

Zhongyu Lu, Weiren Yu, Richong Zhang, Jianxin Li, Hua Wei
State Key Laboratory of Software Development Environment
School of Computer Science & Engineering
Beihang University, Beijing, China
Email: {luzy, yuweiren, zhangrc, lijx, weihua}@act.buaa.edu.cn

***Abstract*—Microblog, such as Weibo and Twitter, has become an important platform where people share their opinions. Much research has been done to detect topics and events in microblogs. Due to the dynamic nature of events, it is more crucial to monitor the evolution and trace the development of the events. People pay more attention to the whole evolution chain of the events rather than a single event. In this paper, we propose a method to automatically discover event evolution chain in microblogs based on multiple similarity measures including contents, locations and participants. We build a 5-tuple event description model specifically for events detected from microblogs and analyze their relationships. Inverted index and locality-sensitive hashing are used to improve the efficiency of the algorithm. Experiment shows that our method gain a 143.33% speed up against method without locality-sensitive hashing. In comparison with the ground truth and a baseline method, the result illustrates that it effectively covers ground truth and outperforms the baseline method especially in dealing with the long-term spanning events.**

***Index Terms*—Event Evolution, Microblog, Similarity Measure, Inverted Index, Locality-Sensitive Hashing.**

I. Introduction

With the rapid development of social media, particularly Microblog, has become prevalent all over the world. Weibo and Twitter are typical representatives of Microblog service. People share their lifestyles, comment on breaking news and follow trending events on Microblog, which makes it possible to detect and monitor real-life events among these user-generated microblog contents.

Existing event detection techniques focus on bursty keywords detection[1] and clustering[2] or topic model[3]. Most of them represent events with a set of detected keywords. However, real-life events are evolving over time and it is more crucial to monitor the evolution and trace the development of those events. Not only the newest events but also the dynamics of these events are users' interests. This fact suggests that people pay more attention to the whole evolution chain of the events rather than a single event.

To the best of our knowledge, there are some research mining event evolution pattern on news corpora. Story link detection(SLD)[4] is a core task in traditional topic detection and tracking(TDT) research aiming at discovering relationships among documents. These methods give solutions to construct links for long-text documents and do not quite fit for short-text microblogs.

To overcome the limitation of existing approach, we propose a method to build event evolution chain automatically for Weibo based on multiple similarity measures in this paper. It is designed for the scenario where users need to obtain evolution processes of either long-term or short-term events from a huge amount of events accumulated from short-text event detection system. The main contributions of this paper are listed as follows:

- We introduce a novel 5-tuple model based on event attributes, including time, location, participants, description and related posts, for events detected from microblogs.
- We develop an event similarity measure function to evaluate events dependencies. An event evolution chain is built based on the dependency evaluation result.
- We improve the efficiency of our algorithm by two-layer filtering based on inverted index and locality-sensitive hashing(LSH)[5]. Result shows a 143.33% speed up against method without LSH.
- Our experiments on a manually annotated dataset also show the effectiveness of our method in comparison with a baseline method on both precision and recall.

The rest of this paper is organized as follows: Section II reviews related works. Section III introduces our model and algorithm. Section IV describes experimental results. Section V concludes the paper.

II. Related Works

In this section, we describe related works about event evolution and similarity measures. Relevant works of discovering events relationship can be classified into two categories:

Offline Correlation Building Methods for Existing Events. Makkonen[6] builds a multi-vector event model to describe events and evaluate event relationships by calculating the similarity of these vectors. However, it treats the different vectors equally and doesn't distinguish the contributions of terms, locations, names and temporals. Yang et al.[7] define an event evolution scoring function to estimate the likelihood of the existence of evolution relationship. Although this function integrates event content similarity with temporal proximity and document distributional proximity, it doesn't take event attributes, such as event location, participants, into consideration. Nallapati et al.[8] test different combinations of multiple features, and find out that average content similarity combining with time decay is the best choice. But all these methods are applied to news corpora and can not be applied directly to short texts. Our method belongs to this category and we design an

978-1-4799-8937-9/15 $31.00 © 2015 IEEE
DOI 10.1109/HPCC-CSS-ICESS.2015.81

events similarity score function specifically for microblog in this paper.

Online Event Tracking Methods for Streaming Texts. Lee et al.[9] propose a method to build story-teller for streaming social content, which constructs a post graph and dynamically detect (k,d)-core subgraphs to represent events with the time window sliding. It builds the evolution relationships by monitoring the development of subgraphs. This method can only get the dependencies of events in the adjacent windows and is not applicable for events spanning over a long term. Lin et al.[10] use dynamic pseudo relevance feedback to obtain relevant tweets and generate event storylines via graph optimization. But it needs a keywords query given by users, and the effectiveness of result depends on whether those keywords cover the event well. Therefore this method does not fit for our problem without any priori knowledge.

The core problem of discovering relationship between two events is building similarity metrics. In our method, we adopt metrics including entities, textual features, locations and participants to evaluate event relationships.

III. Model and Algorithms

In this section, we introduce our method for building event model and discovering event evolution chain from microblogs.

A. Preliminaries

Event Definition. We denote an event as a 5-tuple:

$$E =< t, loc, par, desc, posts > \quad (1)$$

where t is the timestamp when an event was detected. loc is the location where the event happened. par is a set of participants who were involved in this event. $desc$ is the narrative description and summarization of the event, which is usually a short sentence. $posts$ is a set of microblogs that are related to this event, and it has size limitation of 10.

Problem Formulation. Given a latest event E_0, we define the evolution chain of E_0 as a sequential pattern:

$$E_n \rightarrow E_{n-1} \rightarrow \ldots \rightarrow E_1 \rightarrow E_0 \quad (2)$$

where $E_i \rightarrow E_j$ means event developed from E_i to E_j. This chain traces back the whole development of the event E_0 along the timeline.

B. Event Detection

The focus of this paper is event evolution other than event detection algorithm. So we use the state-of-the-art event detection algorithm to discover events for our work of event evolution chain generation. We detect events from microblog streams with two steps: trending keyword detection and event detection. The first step detects trending keywords using the method proposed by Yu et.al.[11]. In the second step we adopt a overlapping community detection method[12] to find out events from these trending keywords. The event is represented with a set of keywords. The details of the method is beyond the scope of this paper.

After finishing above steps, we get events which are represented by keywords. However, the information in these keywords is insufficient and it is also difficult for users to understand. To get more informative comprehension of the events, we need to fill the 5-tuple for each event. Firstly the time when the event was detected is set as t. Secondly we track down related microblogs containing all these keywords by querying microblog index and put them into set $posts$. It is possible that $posts$ be empty and we simply discard these events. Thirdly we obtain event summarization $desc$ from $posts$ through Algorithm 1. The main idea of this algorithm is to get the hashtag or sentence containing most of the event keywords because they are easier for the users to understand. Finally we tokenize these microblogs in $posts$ and find out named entities including the names of location, people and organizations. The most mentioned location is set as loc while people and organizations are put into set par. Null loc and empty par are permitted in this model.

Algorithm 1 Event Summarization

```
for each p ∈ posts do
  if p contains hashtag then
    return hashtag;
  end if
end for
Result = null, Flag = 0;
for each p ∈ posts do
  Split p into sentences by punctuation, get Set S;
  for each s ∈ S do
    Calculate the number of event keywords in s: num;
    if num > Flag then
      Result = s;
      Flag = num;
    end if
  end for
end for
return Result;
```

TABLE I
An Example for Event Model

Event Keywords		FAW Group, Inspection Group,Investigate, Central Commission for Discipline Inspection, Board Chairman, Illegal, Undisciplined
Event Model	t	2015/3/15 18:10
	loc	Null
	par	Xu Jianyi, FAW Group, Zhu Baocheng, Ministry of Supervision
	$desc$	The board chairman of FAW Group was investigated by CCDI.
	$posts$	CNSTOCK: FAW Group chairman and Party secretary Xu Jianyi is being investigated on suspicion of violating Party discipline and national laws, according to a government statement released on March 15 on CCDI. ...

This event model depicts events from five aspects, which

is informative and user-friendly. An example is illustrated in TABLE I. It is about Xu Jianyi, the board chairman of FAW Group, who was investigated by the Central Commission for Discipline Inspection because of corruption.

C. Event Relationship Evaluation

To build the event evolution chain, the main challenge is how to evaluate the relationships between two events. Given two events E_i and E_j, which are separately denoted as $< t_i, loc_i, par_i, desc_i, posts_i >$ and $< t_j, loc_j, par_j, desc_j, posts_j >$ using our event model, we define the event similarity score function as:

$$\begin{aligned} Sim(E_i, E_j) = \alpha \cdot Sim_{posts}(posts_i, posts_j) + \\ \beta \cdot Sim_{loc}(loc_i, loc_j) + \gamma \cdot Sim_{par}(par_i, par_j) \end{aligned} \tag{3}$$

where Sim_{posts}, Sim_{loc} and Sim_{par} denote similarity measures of related microblogs, location and participants respectively. α,β,γ are weight coefficients of these measures and can be adjusted with the requirement of $\alpha + \beta + \gamma = 1$.

Sim_{posts} evaluates the similarity of events' related posts, which is calculated using the average cosine similarity of each microblog pair of the two events. We represent each micoblog as a weighted term vector $p_i =< w_{i,1}, w_{i,2}, \cdots, w_{i,n} >$. $w_{i,j}$ denotes the frequency of term j in p_i and n is the size of vocabulary. Sim_{posts} is defined as follows:

$$cos\_sim(p_i, p_j) = \frac{\sum_{k=1}^{n}(w_{i,k} \cdot w_{j,k})}{\sqrt{[\sum_{k=1}^{n}(w_{i,k})^2]} \cdot \sqrt{[\sum_{k=1}^{n}(w_{j,k})^2]}} \tag{4}$$

On the basis of Equation (4), Sim_{posts} is defined as:

$$\begin{aligned} Sim_{posts}(posts_i, posts_j) = \\ \frac{\sum_{p_k \in posts_i} \sum_{p_l \in posts_j} cos\_sim(p_k, p_l)}{|posts_i| \cdot |posts_j|} \end{aligned} \tag{5}$$

It's assumed that events happen in the same place are more possible to have evolution relationships. Sim_{loc} is employed to evaluate the similarity of events' location:

$$Sim_{loc}(loc_i, loc_j) = \begin{cases} 1 & if\ loc_i\ equals\ loc_j; \\ 0 & otherwise. \end{cases} \tag{6}$$

Just like location, same participants also reflect the relationships among events. We define $Sim_{par}(par_i, par_j)$ as the Jaccard Coefficient of set par_i and set par_j:

$$Sim_{par}(par_i, par_j) = \frac{|par_i \cap par_j|}{|par_i \cup par_j|} \tag{7}$$

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

$$E_i \rightarrow E_j\ iff.\ Sim(E_i, E_j) > \varepsilon\ and\ t_i < t_j. \tag{8}$$

We tune ε to 0.4 in this paper through experiments on a manually annotated dataset.

D. Event Evolution Chain Generation

After introducing the method to build the dependency relationship between two events, we discuss how to build the whole evolution chain for given event E_0.

1) Trace back the whole evolution chain: Since our evolution model is a chain, we can generate it iteratively. During each iteration, the algorithm finds out an event E' for input event E meeting the requirement: $E' \rightarrow E$. The output E' is regarded as the input event for next iteration. This process comes to an end if no related event is found for input event. In consideration of the amount of events, we adopt a two-layer filtering method to decrease the computation of similarities between input event and every earlier event detected by our system during each iteration.

Inverted Index Filtering. We assume that related events share at least one noun in their keywords sets. Given input event E in each iteration, we choose nouns from event keywords of E . An inverted index is built to map nouns to events. And events possibly related with E and happened before E are retrieved from inverted index using these nouns. These events form Possible Set PS. This step filters most irrelevant events out but the size of PS is still relatively large.

LSH Filtering. Locality-Sensitive Hashing(LSH) was proposed to solve the Approximate Nearest Neighbors(ANN) problem. It uses hash functions families to map similar items to the same buckets with high probability. In this paper, we adopt LSH to get a small set of events similar to the given event to reduce the number of comparisons. Events in set PS and input event E are represented as weighted noun vector $E_i =< w_{i,1}, w_{i,2}, \cdots, w_{i,m} >$, where $w_{i,j}$ is calculated by:

$$w_{i,j} = \begin{cases} 1 & if\ keywords\ set\ of\ E_i\ contains\ noun\ j; \\ 0 & otherwise. \end{cases} \tag{9}$$

m is the size of our noun vocabulary. We use cosine distance hash family to build our hashing model. And the number of hashes and hash tables are both set as 5 after our experiments on a annotated dataset. The 10 nearest events of E are obtained from LSH and put into Candidate Set CS, which is prepared for further process.

After the two-layer filtering, we've got a candidate events set CS whose maximum size is 10 for input event E. It's worth mentioning that all these events have happened before E because of PS's characteristic. We sort events of CS by time descending order, and then compute similarity between input event E and each event in CS one by one. The first event E' meets the requirement of Equation (8) is the output event of this iteration round.

2) Merge the same events: After the previous steps, there may be same events appearing in the evolution chain because same events are detected continuously in the event detection module for their long-term span. We adopt a threshold ε' set as 0.8 by default to merge them.

Algorithm 2 illustrates the method to generate event evolution chain for given event E_0. An example is given in TABLE

TABLE II
AN EXAMPLE FOR EVENT EVOLUTION CHAIN

Time	Event description
2015/1/2 11:50	The Governor of Heilongjiang province Lu Hao emphasizes the importance of maintaining non-gmo soybean.
2014/12/24 12:20	An agricultural official says genetically modified foods having obtained safety certificate are safe.
2014/12/23 10:10	China approves for importing Dupont genetically modified high oil soybean.
2014/11/15 18:10	The national development and reform commission considers allowing foreign research and development of genetically modified in China.

II. This event evolution chain describes the development process of importing genetically modified soybean in China.

Algorithm 2 Event Evolution Chain Generation

Input: E_0
Output: Array $Chain[]$
$E = E_0$;
repeat
  $E' = null$;
  Get events set PS by searching event index;
  Get events set $CS \subset PS$ by LSH filtering;
  Convert CS to events array CA and sort CA by time descending order;
  for $i = 1$; $i \leq CA's\ size$; $i++$ **do**
    Calculate $Sim(E, CA[i])$;
    if $Sim(E, CA[i]) > \varepsilon$ **then**
      $E' = CA[i]$;
      break;
    end if
  end for
  Insert E to the beginning of $Chain$;
  $E = E'$;
until $E == null$;
//Merge the same events
$len = Chain's\ size$;
for $i = 1$; $i < len$; **do**
  if $Sim(Chain[i], Chain[i-1]) > \varepsilon'$ **then**
    Remove i from $Chain$;
    $len--$;
  else
    $i++$;
  end if
end for
return $Chain$.

IV. EXPERIMENTS

In this section, experiments are designed to test our method from both efficiency aspect and effectiveness aspect.

A. Settings

Dataset: We started collecting Weibo data and detecting events in Sep.18.2014. Until now there has been about 1.7 billion microblogs collected through Weibo API and about 1 million events detected. Our vocabulary contains 201,196 words and noun vocabulary 134318 words.

Parameters: The parameters α, β, γ, ε, ε', n_hashes, n_tables, $n\_neighbours$ used in this paper are listed in TABLE III.

TABLE III
PARAMETERS SETTING

Parameter	Default	Description
α	0.7	coefficient of posts similarity
β	0.1	coefficient of location similarity
γ	0.2	coefficient of participants similarity
ε	0.4	threshold of evolution relationship
ε'	0.8	threshold for merging events
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

Implementation: We use *ICTCLAS*[1] to tokenize microblogs and recognize named entities. LSH is implemented by *TarsosLSH*[2]. Microblogs and events are stored in HBase and indexed by Elasticsearch. Our algorithm is implemented in Java with JDK 1.7.

Platform: HBase and Elasticsearch are set up on a cluster of 8 machines, each with 2 Xeon E5-2650 CPUs, 256GB RAM and Debian7 64-bit operation system. Our algorithms are conducted on a machine with Intel 3.20GHz CPU, 8GB RAM, running with Windows 7 64-bit operation system.

B. 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 Algorithm 2.

Fig. 1 shows the distribution of event evolution chains' length produced by our algorithm. We can see that 83% events have a single-event chain, which means they have no previous related events. It's reasonable since most of the events in the social media develop quickly and vanish quickly. Apart from these events, 141 out of the rest 170 events have evolution chains whose length spans from 2 to 5, which means that evolving events have no more than 5 phases in most cases.

Fig. 2 shows the comparison of average time spent on generating evolution chain between our method with and without LSH filtering step. We can see that the average time of all 1000 events is 1.67s with LSH and 5.04s without LSH. When only events whose chain lengthes exceed 1 are taken into consideration, the time rises to 4.92s and 10.28s. To evaluate the algorithm more precisely, we calculate the average time of each iteration, i.e. the time to produce a "$E_i \rightarrow E_j$" relationship, which is 1.20s with LSH and 2.92s without LSH. In other words, the LSH filtering step improves the computing speed by 143.33%.

[1]http://ictclas.nlpir.org/
[2]https://github.com/JorenSix/TarsosLSH

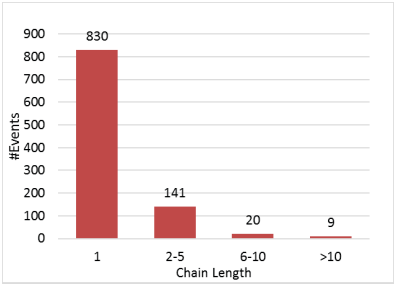

Fig. 1. Distribution of Evolution Chains' Length

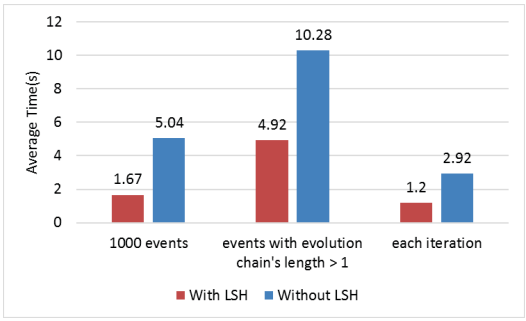

Fig. 2. Comparison of Average Time on Generating Evolution Chain

C. Effectiveness

In this part we evaluate the effectiveness of our algorithm by comparing the algorithm with a baseline method and ground truth generated from news.

Ground Truth. We manually choose 20 input events from the previous mentioned 1000 events. These events all have obvious evolution processes in real world and we can find out corresponding news reports of each development phase for them. To generate baseline event chain, we choose a news title as description for each phase and manually combine them into an evolution chain for each event.

Baseline. Some works also use the combination of multiple features to measure the relationships between two events[7][8][13][14]. Content similarity and temporal proximity are most widely used and proved to be significant on news corpora[8]. Therefore we adopt the combination of content similarity and temporal proximity as the baseline. It is computed as follows:

$$Sim(E_i, E_j) = Tp(E_i, E_j) \cdot Sim_{posts}(posts_i, posts_j) \quad (10)$$

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

$$Tp(E_i, E_j) = e^{-\frac{\alpha|t_i - t_j|}{T}} \quad (11)$$

T is the time distance between the earliest and the latest event in our system. α is the time decay factor and is set as 1 in this experiment. We just replace the $Sim(E_i, E_j)$ in the baseline method and the rest algorithm remains the same as ours.

Evaluation. We use the following traditional metrics **Precision**(P), **Recall**(R) and **F1-Measure**(F1) to quantize the effectiveness of our algorithm. They are computed as follows:

$$P = \frac{|G \cap C|}{|C|}; \quad (12)$$

$$R = \frac{|G \cap C|}{|G|}; \quad (13)$$

$$F1 = \frac{2 \cdot PR}{P + R}; \quad (14)$$

where G is the set of events in the ground truth dataset, C is the set of events in output result generated by each one of the different methods including baseline method and our methods with various parameters.

TABLE IV 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 $\alpha = 0.7$, $\beta = 0.2$, $\gamma = 0.1$ has the highest recall and the setting $\alpha = 0.7$, $\beta = 0.1$, $\gamma = 0.2$, which is the default setting, has the highest precision and F1, and outperforms the baseline on both precision and recall. Particularly, comparing our default-setting method with baseline method, the difference of **Recall**s is much larger than **Precision**s. We check the result and find out that this is mainly because Tp in Equation 10 filters out events having a big time gap with the input event in each iteration. But sometimes a real-life event may have different development phases spanning over several weeks, months or even years. For example, a crime case may spans several years from crime happening to first trial, second trial and execution. In our method time is used to determine the chronological order of the events in evolution chain and does not influence the similarities among events.

TABLE IV
EFFECTIVENESS RESULT

	α	β	γ	P	R	F1
Our Method	1.0	0.0	0.0	0.4455	0.6818	0.5389
	0.9	0.1	0.0	0.4302	0.5606	0.4868
	0.9	0.0	0.1	0.5000	0.6212	0.5541
	0.8	0.1	0.1	0.3960	0.6061	0.4790
	0.8	0.0	0.2	0.3929	0.5000	0.4400
	0.7	0.2	0.1	0.3529	**0.7273**	0.4752
	0.7	0.1	0.2	**0.5422**	0.6818	**0.6040**
	0.6	0.2	0.2	0.2368	0.6818	0.3516
	0.6	0.1	0.3	0.2500	0.5606	0.3458
Baseline Method				0.5000	0.4545	0.4762

Case study. TABLE V shows the comparison between our method and baseline method based on an example about a notorious crime case happened in Fudan University where the suspect was accused of poisoning his roommate to death. The ground truth describes it with three evolution phases focusing on the second trial. For baseline method, it doesn't find out the evolution phases of the given event as the time span is so large

TABLE V
CASE STUDY

Ground Truth	Our Method	Baseline Method
2015/01/08 The court maintains the former death sentence on attempted murder in the second trail of Fudan poisoning case. **2014/12/08** Fudan poisoning case's second trial is held in 10am today. **2014/12/01** The suspect of Fudan poisoning case writes an apology letter to the victim's parents.	**2015/01/08** The court maintains the former death sentence on attempted murder in the second trail of Fudan poisoning case. **2015/01/08** The court will pronounce judgement of the 2^{nd} trial today and victim's father hope to maintain the death penalty. **2014/12/08** LIVE: Defendant of Fudan poisoning case cries in court. **2014/12/08** Fudan poisoning case's second trial will be held in 10am today, victim's parents will be in court. **2014/12/01** The suspect of Fudan poisoning case writes an apology letter to the victim's parents. The 2^{nd} trial will be held on 8th December.	**2015/01/08** The court maintains the former death sentence on attempted murder in the second trail of Fudan poisoning case.

that the events similarity is diminished by temporal proximity prominently. For our method, it covers all the three phases of this long-term event and describes it with a finer granularity. For example, the second phase in the ground truth is split into two phases in our method, one describes the schedule of the second trial and one depicts the live situation. A finer granularity means $|C|$ is bigger than $|G|$ in Equation (12) and (13). This is the reason why **Recall** of our method is higher than **Precision**. This case explains our method's superiority over the baseline.

V. CONCLUSION

In this paper, we define a new description model for events detected from Weibo and describe the algorithm to generate event evolution chain automatically. We combine the similarity measures of related microblogs, locations and participants to evaluate the relationships among events. Considering the amount of accumulated events(about 1 million events in our system until now), it's impossible to calculate given event's similarity comparing to each detected event. We adopt a two-layer filtering method to preprocess the events and find out a candidate events set with a small size for each given event. Our experiments shows the high performance of our method from both efficiency aspect and effectiveness aspect. For efficiency, our method can generate an event evolution chain for a given event in less than 2s, which means a 143.33% speed up against method without LSH. For effectiveness, our method outperforms baseline on both precision and recall for its better result in dealing with long-term events. The case study also shows that our method produces an evolution chain covering ground truth well and having a finer granularity.

In future work, we plan to integrate heterogeneous network analysis into our model to obtain richer information about events and explore more event similarity measures.

ACKNOWLEDGMENT

This work is supported by grants from the China 973 Fundamental R&D Program(No.2014CB340300), NSFC Program (No. 61472022), and SKLSDE-2014ZX-04.

REFERENCES

[1] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 1155–1158.

[2] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava, "Dense subgraph maintenance under streaming edge weight updates for real-time story identification," *Proceedings of the VLDB Endowment*, vol. 5, no. 6, pp. 574–585, 2012.

[3] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 1445–1456.

[4] F. Chen, A. Farahat, and T. Brants, "Multiple similarity measures and source-pair information in story link detection." in *HLT-NAACL*. Citeseer, 2004, pp. 313–320.

[5] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.

[6] J. Makkonen, "Investigations on event evolution in tdt," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 student research workshop-Volume 3*. Association for Computational Linguistics, 2003, pp. 43–48.

[7] C. C. Yang, X. Shi, and C.-P. Wei, "Discovering event evolution graphs from news corpora," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 39, no. 4, pp. 850–863, 2009.

[8] R. Nallapati, A. Feng, F. Peng, and J. Allan, "Event threading within news topics," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 2004, pp. 446–453.

[9] P. Lee, L. V. Lakshmanan, and E. Milios, "Cast: A context-aware story-teller for streaming social content," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 789–798.

[10] C. Lin, C. Lin, J. Li, D. Wang, Y. Chen, and T. Li, "Generating event storylines from microblogs," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 175–184.

[11] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, "On anomalous hotspot discovery in graph streams," in *IEEE International Conference on Data Mining (ICDM)*, 2013, pp. 1271–1276.

[12] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[13] D. Huang, S. Hu, Y. Cai, and H. Min, "Discovering event evolution graphs based on news articles relationships," in *e-Business Engineering (ICEBE), 2014 IEEE 11th International Conference on*. IEEE, 2014, pp. 246–251.

[14] R. Ahirrao and S. Patel, "An overview on event evolution technique," *International Journal of Computer Applications*, vol. 77, no. 10, pp. 7–11, 2013.