Ring: Real-Time Emerging Event Monitoring over Text Streams

Weiren Yu, Jianxin Li, Richong Zhang, Shuai Ma, Jinpeng Huai
SKLSDE Lab, Beihang University, Beijing, China
{yuwr, lijx zhangrc}@act.buaa.edu.cn, {mashuai, huaijp}@buaa.edu.cn

***Abstract*—We demonstrate RING, a real-time emerging event monitoring system over microblog text streams. RING integrates our efforts from both event monitoring research and system research. From an event monitoring perspective, adopting a graph analytics approach, RING (1) detects emerging events at an early stage even with small magnitude of traffic (2) is the first to discover multi-aspect hierarchical sub-events of emerging events in a streaming setting (3) reveals evolution of events at multiple time granularities from hours to months. From a system perspective, RING (1) is able to process the whole Weibo or Twitter text stream with linear horizontal scalability (2) optimizes time-ranged keyword query for improved efficiency of both event evolution tracking and user query (3) implements its graph stream model on Spark and improves the incremental update efficiency of Spark platform. It also comes with friendly user interface to visualize events and statistics.**

I. INTRODUCTION

Emerging event monitoring over microblog platforms attracts much attention from both research and application domains due to the real-time nature and viral diffusion of information. In fact they have many times been the first reporter or even the major hosting venue of significant events, such as earthquake forecast or a presidential election campaign. A wide variety of events would emerge from such platforms, ranging from political and daily affairs to natural disasters and public security menace. Not only are the events that attract mass attention important, but also those events with great potential to go viral while only drew attention from a small fraction of users, *e.g.,* a road stabbing with growing witnesses tweeting about it. Despite the versatility in topics and traffic, different aspects of events and different angles of user descriptions would complement each other. Further insights, which we refer to as hierarchical sub-event structure, could be revealed, such as multiple aspects of the event, *e.g.,* the history and current development of a political event, or causality of events, *e.g.,* the capture of a criminal and the crime he had committed, or categorical information, *e.g.,* different genres of news from the same agency. What's more, trends would emerge from different time granularities from minutes to hours, *e.g.,* a concert or sports game, or even days to months, *e.g.,* a publicly concerned long trial. The temporal evolution pattern could be recovered to get the big picture of events. Emerging event monitoring, *i.e.,* early detection, hierarchical correlation analysis, and temporal evolution tracking of these events in real-time, provides valuable insights. Such intelligence are desirable and directly related to government agencies, news groups and marketing strategies, etc.

However, traditional topic modeling methods (e.g. Latent Dirichlet Allocation) and its derivatives could not be directly applied to noisy short texts nor could they perform early detection of emerging events in real-time. TwitterMonitor [1] provide online detection for general trending events but could not reveal multiple aspects of events nor track the evolution of them. CLEar [2] can provide real-time event detection and tracking but could not provide hierarchical analysis of events. Neither of them could identify potential viral events at small scale. Signitrend [3] could detect potential viral events at small scale but confined to only detection and could not provide event tracking and correlation analysis. The method would also tend to generate trends that only contain a single keyword, which is hard to comprehend for users. Another important aspect of event monitoring application is system optimization. None of these above methods or systems provide horizontal scalability with distributed computation nor do they investigate in system optimization for their applications.

In this demonstration, we present RING, a real-time emerging event monitoring system over microblog text streams that integrates our efforts in both event monitoring research and system research. Specifically, we developed event detection, event evolution tracking and event refinement algorithms to monitor events in real-time. We optimizeed full-text indexing engine and distributed processing engine for better efficiency. We also provide friendly visualization to facilitate the analysis of emerging events. The features of RING system is as follows. From an event monitoring perspective, RING provides (1) early detection of potential trending events even with small traffic of tweets, long before they go viral. Trends at different time granularities, *i.e.,* trends over 10 minutes or 1 hour, could be simultaneously revealed at detection time to facilitate better tracking of events. (2) multi-aspect hierarchical view of correlated sub-events, e.g. different aspects of events such as highlights of a sports game, causality of events such as the trigger and outcome of an investigation, or categorical structure of events such as different genres of detected news reports. (3) event evolution tracking to monitor the temporal development of an event and trace of its origin. (4) context information of events and filtering of spam events. From a system perspective, RING provides (1) horizonal scalability to handle full Weibo or Twitter data. (1) optimized indexing for efficient ranged query over time for event tracking and user interaction. (2) optimized distributed processing engine handle large volume of data stream. (3) friendly user interface to visualize events, contexts and evolutions.

RING is the first system to enjoy such rich set event monitoring features with dedicated system optimization efforts.

It is also the first system to provide multi-aspect hierarchical view of correlated sub-events under real-time emerging event monitoring scenario. The capability of RING system is demonstrated through functional features such as top trending events, event evolution analysis, event query and context extraction e.g. geographical information.

II. EMERGING EVENT MONITORING

Event Detection. The event detection method consists of three steps: trending keyword detection, event detection and hierarchical sub-event correlation construction. An event is essentially defined as a set of weighted keywords, enriched by its context information such as detection time, geographical location, participants and so on. Based on co-occurrence relationship of keywords, we adopt a similar outlier detection framework as in [4] for early detection of trending keywords. The incoming text is tokenized and keywords are defined as nodes and their co-occurrence relationship as edges in the graph. The text stream is modeled as streaming edge weight updates where edge frequency based on weighted decay is updated with each pair of keywords co-occurrence. Trends over different time granularities could be monitored simultaneously by setting different decay rate. The graph stream model is fully distributed with a linear scalability since it follows a data-parallel paradigm where the data could be partitioned according to keywords, *i.e.,* each computing node would only need data containing the keyword in its partition. An efficient context statistics maintenance strategy is developed for efficient update. Specificallys, for each node i three pieces of information are maintained: (i) the last time stamp $L(i)$ at which an edge was received of node i (ii) the set of nodes in $S(i,t)$ (iii) an array of frequency values $F(i,j,L(i))$ for each node $j \in S(i,t)$.

A scalable overlapping community detection method is applied to detect events over trending keywords. Keywords of the same event would co-occur and be more densely linked internally than with keywords from a different event. It would be intuitive to follow modularity-based [5] or betweenness centrality based [6] community detection methods for event detection. However, such methods would lead to 4 major drawbacks: (1) non-local: density changes in parts of the graph would affect the overall result for community detection. (2) mixed contexts: a keyword cannot appear in different events (3) co-occurrence by chance: keywords in different contexts would likely to be assigned to the same event due to lack of explicit definition of keyword community (4) no hierarchy: hierarchical correlation analysis is absent for such methods. Events would naturally consist of different aspects which formulate a hierarchical structure.

To distill events from these aspects and get the bigger picture, we use *k-clique percolation* to find overlapping communities as events [7]. A *k-clique-community* is defined as a union of all k-cliques that can be reached from each other through a series of adjacent k-cliques, where adjacency means sharing *k-1* nodes. Such definition of community enjoys several advantages versus existing methods: (1) local definition of community that would not be affected by change in parts of the graph (2) natural overlap to decode polysemy and diverse contexts (3) reduce co-occurrence by chance through parameter k (4) hierarchical view of sub-events is encoded in the maximal cliques of the communities.

A frequent pattern tree mining technique is adopted to retrieve hierarchy of correlated sub-events. Valuable insights, such as different aspects of events, causality of events or categorical structure of events, could be revealed. We are the first to provide a hierarchical view of correlated sub-events for the emerging event detection scenario. Highly matching tweets related to events are retrieved as representation texts through querying search engine with keywords of sub-event.

We manually inspect 100 detected real world events and its first related tweet in our dataset. We find that the method can detect 79% of events within 10 minutes and 40% within 5 after the first tweet appeared. The method can outperform state-of-the-art topic and event detection methods based on keyword co-occurrence [5], [6], [8] in terms of keyword coherence (NPMI) and summarization quality (ROUGE-1). The method is implemented on Spark and share a cluster of 8 machines with other algorithms and system components, where it can handle 16k tweets per second and bear linear scalability given more machines.

Event Refinement. The refinement procedures are responsible for spam filtering and context enrichment. Based on the method [9], we first remove spam accounts from our data crawler, who are either constantly publishing ads content or actually manipulated by intelligent software for propaganda purposes. We apply a 3-class Naive Bayes classifier to detected trending events which differentiates news, ads and widsom words, where the latter two are major types of spam information on Weibo. We train the classifier with manually labeled data based on features of content, users and temporal information. We further extract the location of the detected events using a location vocabulary and find out the candidates of events with *ICTCLAS*[1], including nouns, people and organizations etc.

Event Tracking. Event tracking aims to trace back the development of events along the timeline. Given a latest event E_0, we define the evolution chain of E_0 as:

$$E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_n \rightarrow E_0 \tag{1}$$

where $E_i \rightarrow E_j$ means event developed from E_i to E_j. We first get an event candidate set CS for E_0. Under the assumption that related events share at least one noun in their keyword sets, CS is retrieved using nouns in the keyword set of E_0. An inverted index is built to map nouns to events. We then build the event chain of E_0 through a set of similarity metrics for the location loc_i, participant set ps_i, and event related post set ws_i of event e_i in CS. We define the similarity of two events e_i and e_j as

$$\begin{aligned} Sim(e_i,e_j) = \alpha \cdot Sim_{ws}(ws_i, ws_j) + \\ \beta \cdot Sim_{loc}(loc_i, loc_j) + \gamma \cdot Sim_{ps}(ps_i, ps_j) \end{aligned} \tag{2}$$

$Sim_{ws}(ws_i, ws_j)$ is based on average cosine similarity of related posts. $Sim_{loc}(loc_i, loc_j)$ equals 1 if e_i and e_j share the same location and 0 otherwise. $Sim_{ps}(ps_i, ps_j)$ is based on Jaccard similarity of participant sets ps_i and ps_j. $\alpha+\beta+\gamma = 1$ and can be adjusted empirically. We calculate $Sim(e_i, E_0)$ for each event in CS and note set $Chain$ for E_0: $Chain =$

[1] http://ictclas.nlpir.org/

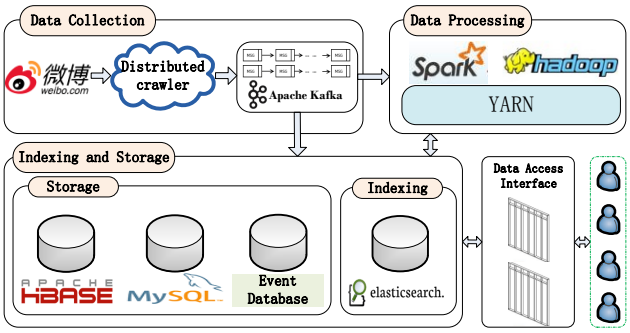

Fig. 1. The Architecture of RING System

$\{e_i \in CS | Sim(e_i, E_0) > \varepsilon\}$. Threshold ε is used to filter out potential unrelated event candidates. It is worth mentioning that we use another threshold ε' to merge near duplicate events in $Chain$. ε' is larger than ε and smaller than 1. We tune ε and ε' empirically on a manually annotated dataset.

This method [10] leverages locality sensitive hashing and time range query optimized indexing to improve its efficiency It is effective in revealing the evolution of both the long-term and short-term events, rather than only recent events [11], [12].

III. THE RING SYSTEM

As shown in Figure 1, RING mainly consists of three modules, namely data collection, data indexing & storage and data processing. For *data collection*, we developed a distributed crawler to continuously fetch microblogs through Weibo API[2]. The collected data is forwarded to indexing and processing modules through Kafka[3] to decouple their dependency. For *indexing and storage*, RING utilizes HBase[4] and elasticsearch[5] for data storage and full-text indexing, which is able to process large volumes of real-time microblog stream. We further optimize elasticsearch over time range query for event tracking and query applications. For *data processing*, we implement our models and algorithms on Spark[6] and design an incremental update optimization strategy to improve distributed processing efficiency.

Data Collection. We built a distributed crawler to collect data from Weibo, the largest microblog platform in China. The crawler continuously collects the latest microblogs published by users preferrably with a large number of followers, *i.e.,* opinion leaders. The crawler has a master/slave architecture. The master node utilizes key/value store to perform task scheduling. Slave nodes get the assignments from the master node and crawl data from Weibo. A task would monitor the reposts and comments of an original tweet and retrieve the repost and comment list, from which we can construct the forward graph of each tweet message. The tasks are scheduled according to posts' priority, which is weighed by the number of reposts and comments. Each task has a life cycle so that each thread of post can be effectively monitored or terminated. In order to make full use of computer resources and the limited API bandwidth, we deploy the crawler in KVM virtual machines. The data is crawled using Weibo API in public privilege. Due to recent change on the public API, up to now we are able to collect 3 - 10 million microblogs daily.

[2]http://open.weibo.com/wiki/Api
[3]http://kafka.apache.org
[4]http://hbase.apache.org/
[5]https://github.com/elastic/elasticsearch
[6]http://spark.apache.org

Fig. 2. Features of Demonstration

Indexing: We built our full-text index for microblog texts and the emerging events with elasticsearch. Event evolution tracking application issues extensive amount of keyword queries to retrieve related tweets for similarity computation. Each query is issued toward data in a specific time interval as defined in the event detection algorithm. We add a time range structure in the multi-layer index of elasticsearch to directly optimize time range query performance [13]. Such structure for time ranged queries avoids traversing each layer and allows efficient navigation to records within a specified time interval.

Distributed Processing: Our event monitoring algorithm is implemented on Spark. As discussed in Section II, a continuously changing graph of keywords and their co-occurrence relationship is built from the text stream. To improve the efficiency of graph update, we created an index in each partition of the Spark RDD to rapidly locate data records and enable fine-grained incremental updates [14]. This alleviates the overhead of data copy and shuffling when updated nodes only account for a small portion of the graph and when updated statistics of each node can be computed from a known portion of data, *i.e.,* tweets containing certain keywords in our case. Experiments show that the optimization has a speedup of 5.4X for incremental PageRank and 3.7X for statistics update.

IV. DEMONSTRATION

User Interface. We now introduce three function features to demonstrate emerging event monitoring as shown in Figure 2.

Trending Event Ranking: Events detected by RING system are tracked based on the method mentioned in Section II. Events are ranked according to the volume of their related microblogs. The "hottest" trending events in the past 5 hours, 1 day and 1 week are displayed on the front page of RING system. Our monitoring application could cover emerging events

TABLE I. A CASE OF EVENT EVOLUTION CHAIN

Time	Event description
2014-12-01 18:40:00	The suspect of Fudan poisoning case writes an apology letter to the victim's parents. | The 2^{nd} trial will be held.
2014-12-08 08:10:00	Fudan poisoning case's second trial will be held in 10am today, victim's parents will be in court.
2014-12-08 12:50:00	LIVE: Defendant of Fudan poisoning case cries in court.
2015-01-08 07:40:00	The court will pronounce judgement of the 2^{nd} trial today and victim's father hope to maintain the death penalty.
2015-01-08 10:30:00	The court maintains the former death sentence on attempted murder in the second trail of Fudan poisoning case.

TABLE II. A CASE OF HIERARCHICAL SUB-EVENT

Poisoning Case, Fudan	Apologize, Lin Senhao, Open a Court Session, Write
	Scheduled to, Shanghai

ahead of major news sites, *e.g.,* sina.com.cn or baidu.com. Spam information has been effectively removed and missing events are majorly due to absence of data.

Event Analysis: Navigated from a search or trending event ranking, the event analysis page shows the detailed information of a detected event which includes representative tweet as human-readable abstract, time of detection, keyword summarization, participants, event class as news or spam, location, sentiment bias and popularity. The evolution the current event is displayed on a timeline interface, with related microblogs, sub-event hierarchy featuring trending keywords. A word cloud of frequent keywords in related tweets is also displayed.

Geographical Distribution: The geographical distribution page shows a map with ballon markups indicating the number of events in the location. Click on the ballons would display events in a specified location. A search box allows users to query and navigate to the location more quickly.

Case Study. From this case study we demonstrate the effectiveness of RING for multi-granularity, multi-aspect event monitoring that it can tell the story of an event automatically. An event's evolution chain distilled by RING is shown in Table I. Given (by search or trending event ranking interface) the "latest" event in the table, an evolution chain that spans several month was revealed. These events are about the second trial of the infamous poisoning case in Fudan University, China where the suspect poisoned his roommate to death in April 2013. The detected events are all significant in its development and we can see that forecast, process and judgement of the trial all drew a lot of attention, hence detected by RING. The descriptions of these events are extracted automatically either directly from tweets or from representative tweets that contained predefined brackets indicating proper titles. Recall from II that our system reports trending events in 10-minute intervals. Each of these events in Table I has been repeatedly detected in consecutive time intervals, lasting from 20 minutes to 60 minutes. It is indicated by trending keywords detection algorithm that a consecutive exponential growth of tweets in volume has been detected. The system consolidate such similar events to show only one such event and highlight such event for special attention. We manually inspect the average latency of detection of these events, i.e. the time from the first tweet appearing in our data to the detection time, to be 14.4 minutes, which is shown in the table.

The sub-event hierarchy of the first event in Table I is shown in Table II. There are two aspects of the event. The first is about the suspect writing an apology letter to the victim's parents and the other is that the second trial of the poison case is scheduled to be held in Shanghai, showing different aspects as history and current development of the event.

V. CONCLUSIONS

We demonstrate RING, a real-time emerging event monitoring system over microblog platforms that integrates our efforts from both event monitoring research and system research. RING can detect emerging events, build hierarchical sub-event correlation, trace event evolution and support queries to events and texts. RING's infrastructure is equipped with optimized full-text search engine and distributed processing engine. The system present advantages over existing systems [2], [3], [1] from both the semantic and system perspective.

ACKNOWLEDGMENT

The authors would like to thank ring team members: Zhongyu Lu, Haifei Huang, Chenggen Sun, Bo Wu, Jianfeng Wen and Xue Li for the help in setting up the system.

REFERENCES

[1] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *SIGMOG*, 2010.

[2] R. Xie, F. Zhu, H. Ma, W. Xie, and C. Lin, "Clear: A real-time online observatory for bursty and viral events," *PVLDB Demo*, 2014.

[3] E. Schubert, M. Weiler, and H.-P. Kriegel, "Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds," in *KDD*, 2014.

[4] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, "On anomalous hotspot discovery in graph streams," in *ICDM*, 2013.

[5] J. Weng and B.-S. Lee, "Event detection in twitter." *ICWSM*, 2011.

[6] H. Sayyadi and L. Raschid, "A graph analytical approach for topic detection," *ACM TOIT*, 2013.

[7] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, 2005.

[8] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *WWW*, 2013.

[9] H. Liu, Y. Zhang, H. Lin, J. Wu, Z. Wu, and X. Zhang, "How many zombies around you?" in *ICDM*, 2013.

[10] Z. Lu, W. Yu, R. Zhang, J. Li, and H. Wei, "Discovering event evolution chain in microblog," in *IEEE HPCC*, 2015.

[11] P. Lee, L. V. Lakshmanan, and E. Milios, "Cast: A context-aware storyteller for streaming social content," in *CIKM*, 2014.

[12] A. Saha and V. Sindhwani, "Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization," in *WSDM*, 2012.

[13] H. Huang, J. Li, R. Zhang, W. Yu, and W. Ju, "Liveindex: A distributed online index system for temporal microblog data," in *IEEE HPCC*, 2015.

[14] W. Ju, J. Li, W. Yu, H. Huang, C. Sun, and Z. Lu, "igraph: an incremetal data processing system for dynamic graph," in *SOSP Poster*, 2015.