EventEye: Monitoring Evolving Events from Tweet Streams

Hongyun Cai [#], Zhongxian Tang [*], Yang Yang [#†], Zi Huang [#]
[#] The University of Queensland, Australia
[*] Zhejiang University, China
[†] University of Electronic Science and Technology of China, China
h.cai2@uq.edu.au, kohntzx@gmail.com, dlyyang@gmail.com, huang@itee.uq.edu.au

ABSTRACT

With the rapid growth in popularity of social websites, social event detection has become one of the hottest research topics. However, continuously monitoring social events has not been well studied. In this demo, we present a novel system called EventEye to effectively monitor evolving events and visualize their evolving paths, which are discovered from tweet streams. In particular, four event operations are defined for our proposed stream clustering algorithm to capture the evolutions over time and a multi-layer indexing structure is designed to support efficient event search from large-scale event databases. In our system, events are visualized in different views, including evolution graph, timeline, map view, etc. [1]

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

Keywords

Evolving Events Monitoring; Event Indexing; Event Visualization

1. INTRODUCTION

Twitter is one of the most popular online social network service. The dynamically updated tweets provide a wide variety of real-time information which covers a diverse range of real-life events happening all over the world. With the rapid growth of tweets posted everyday, how to efficiently and effectively detect meaningful events and monitor their evolutions to reveal the implicit triggering relationships among events is in high demand.

Event detection in Twitter is usually defined as clustering tweets which share similar textual contents. Although a lot of efforts have been made on social event detection, most of them omit the evolutions of the events. Among them, the single-pass incremental clustering has been widely adopted for event detection in social media [2]. However, it has two limitations. First, it lacks indexing support to the large-scale social data. Second, the evolutions of the events are not monitored. The most similar work to ours is the KeySee system proposed in [3]. KeySee detects events from social stream and tracks the event evolution patterns. They use the sliding time window strategy along with a density-based clustering algorithm to identify events from the data stream. However, the evolution graph is constructed by treating each event snapshot as a node and the trajectory between snapshots as paths. So the evolution of an event is only checked when the time window moves. This may cause losing track of the evolution of highly dynamic events or storing redundant snapshots for steady events.

In this paper, we propose an evolving events monitoring and visualization system, called EventEye, for Twitter. In this system, a Multi-layer Inverted List (MIL) indexing structure is designed to support efficient event manipulation, including event maintaining and event identifying. The traditional single-pass incremental clustering algorithm is extended by taking into account four newly defined event evolution operations to detect events and capture their evolutions over time. The proposed incremental clustering algorithm enables the event detection and monitoring process in real time, which does not require any prior settings such as sliding window size, etc. Users can browse current events or search historical events at any time. To expatiate an event, we adopt different event visualizations including evolution graph, timeline, map view, word cloud, representative tweets and images. Especially, to visualize the evolution of an event, all its relevant events and the associated evolving relationships are illustrated by an evolution graph, where each node represents an event and the edge between any two nodes indicates their evolving relationship.

2. SYSTEM OVERVIEW

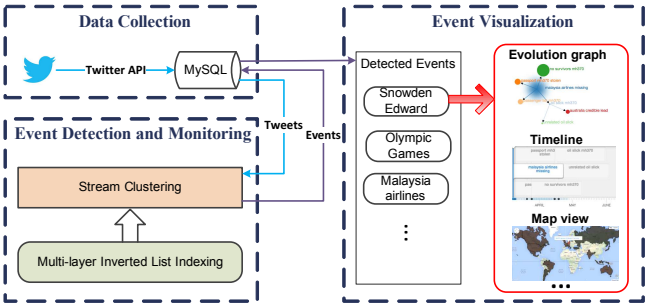

Figure 1: System Architecture

The system architecture is illustrated in Figure 1. Tweets are collected via Twitter API and stored in the database. Our proposed stream clustering algorithm is applied on the collected tweets to detect events and monitor their evolutions. We define four event operations for our clustering algorithm including creation, absorption, split and merge, to capture the event evolution patterns. With the aid of these four operations, the evolutions of the events are recorded in real time. Unlike KeySee which only checks the evolution patterns when sliding window moves, the arrival of any new tweet could trigger split or merge in our system. Specifically, for

[1]The supplementary file is available at https://www.dropbox.com/s/n1pnf0otmlk8jsp/EventEye.pptx

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s).
MM'14, November 3–7, 2014, Orlando, Florida, USA.
ACM 978-1-4503-3063-3/14/11.
http://dx.doi.org/10.1145/2647868.2654871 .

each newly arrived tweet e, its nearest neighbour E_{NN} (the event that has largest similarity between e among all the existing events) is first found. If the similarity between e and E_{NN} is below a predefined threshold θ, a singleton event containing only e is created, otherwise e is absorbed by E_{NN}. After the absorption, if the radius (the smallest similarity from the tweets in E_{NN} to the cluster center) of E_{NN} is smaller than θ, E_{NN} is split into two new events. Once the split is conducted, the newly split events may be merged with nearby events if the merged event satisfies the radius threshold θ. As for the efficiency consideration, MIL indexing structure is proposed as the first event indexing structure to support large-scale dynamically evolving event maintenance and facilitate efficient event search. MIL is a multi-layer indexing structure, which organizes events in different layers guided by different information-specific levels. It inspects most relevant event lists on the lowest layer to avoid exhaustive accesses to longer event lists on the upper layer. After events are detected, users can browse current events or search the historical events at any time. Five views are used to visualize a specified event, including: timeline, evolution graph, map view, word cloud and representative tweets and images. The first three are the major visualization interfaces in our system and will be introduced in the following subsections one by one.

2.1 Timeline of the Detected Events

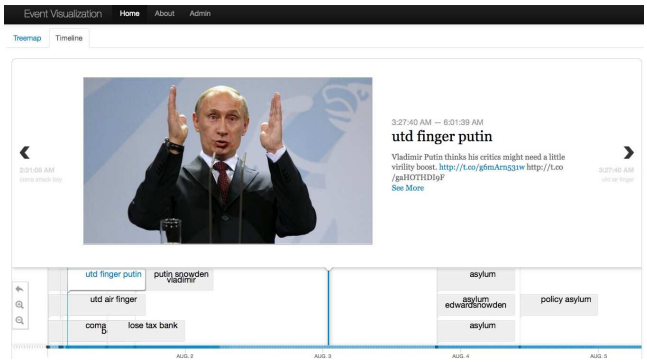

Figure 2: Timeline

To browse the detected events in the system, we design a timeline display of events. One essential characteristic of the social events lies in their temporal orders. Listing events chronologically based on the event starting time gives users a clear view of the event sets. By selecting the start time and the end time in the timeline, events happened within the defined time span will be displayed, with their parallel or sequential relationships along the timeline indicated. Users can also zoom in and out the timeline to view events at different time resolutions. In Figure 2, event "Putin attended a conference" is shown, while all detected events that happened nearby are listed in the timeline.

2.2 Evolution Graph for Event Monitoring

Previous work mostly focus on the detection of the event. Some of them have devoted their efforts on event tracking that trails the details of one specified event in real time. For example, in [1], the authors track the representative tweets of an event and mine geographical diffusion trajectory on the map. Instead of tracking one individual event, we monitor the event evolution among relevant events. The relationships between an event and all its relevant events will be recorded in our system and be illustrated as a graph (e.g., the left diagram of Figure 3), where merge and split operations are clearly displayed. The arrows indicate the event triggering or evolving paths. For example, event "Snowden arrives in Moscow" and event "Putin grants asylum to Snowden" trigger event "Obama cancels vladimir Putin meeting". The timeline is also applied to visualize the event evolutions, where all related events are ordered by their emerging time, so that users can easily get the antecedents and descendant of the events at a glance. By clicking an event on the relationship graph, its hotness value will be shown on the right side of the window (e.g., the bar graph in Figure 3).

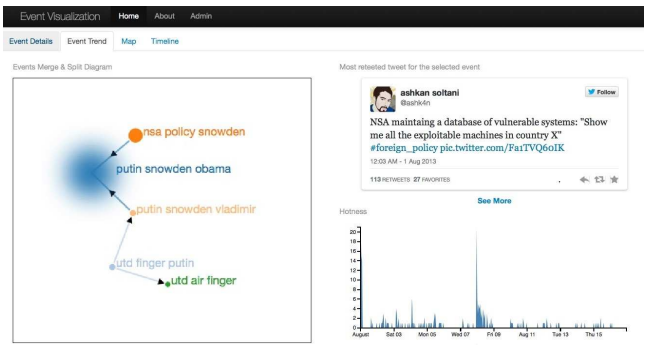

Figure 3: Evolution Graph

2.3 Map View for Event Popularity Analysis

Tweets keep arriving as a spatio-temporal stream. Besides of the timestamp, location also provides important information. Country information is recorded in the database. We analyse and display different popularities in different countries on the map, based on which users get the distribution of the diverse spatial popularities of an event. This distribution provides the answers of some important questions such as whether an event is a regional event or a global event, which countries are interested in this kind of events most, and so on. In addition to the popularity distribution, users can further check the details of this event (e.g., word cloud, representative tweets and images) in each country. This helps users to have a better understanding of the global events. For instance, people from different countries have different opinions on the event that Snowden leaks NSA documents. Various opinions from different countries can be compared in our map view straightforwardly.

3. EVALUATIONS

To evaluate the performance of our system EventEye, we collected 10,681,232 tweets posted from August 1st, 2013 to November 30th, 2013 using Twitter API. We recommend the top 30 hottest events for each day. The experimental results show that EventEye achieves 69.09% precision which outperforms KeySee (62.13%). Furthermore, on average, EventEye only takes around one third of the time to process daily collected data (nearly 90 thousands tweets), i.e., 103.46 seconds vs 352.77 seconds.

4. CONCLUSIONS

We have demonstrated a social event evolution monitoring and visualization system for Twitter. We proposed a novel stream clustering algorithm along with a multi-layer indexing structure to detect social events against large-scale tweets streaming and monitor their evolutions in real time. The events are further visualized with multiple views vividly.

5. REFERENCES

[1] X. Gao, J. Cao, Z. Jin, X. Li, and J. Li. Gesodeck: a geo-social event detection and tracking system. In *ACM Multimedia*, pages 471–472, 2013.

[2] Y. Jie, L. Andrew, C. Mark, R. Bella, and P. Robert. Using social media to enhance emergency situation awareness. *IEEE Intelligent Systems*, 27(6):52–59, 2012.

[3] P. Lee, L. V. S. Lakshmanan, and E. E. Milios. Keysee: supporting keyword search on evolving events in social streams. In *KDD*, pages 1478–1481, 2013.