A Platform For Collaborative Visual Analysis on Streaming Messages

Zipeng Liu Zhenhuang Wang Siming Chen Zuchao Wang Zhengjie Miao Xiaoru Yuan*

Peking University

ABSTRACT

We proposed a collaborative platform to analyze streaming microblog messages in real-time on the emergence of security events, for VAST 2014 Mini Challenge 3. Our team members monitored and analyzed the streaming data simultaneously on it, where we could flexibly distribute workloads by filtering the data streams as we liked, and share notable information such as suspects and locations. With the assistance of a keyword analyzer, we were able to stay vigilant for potential emergencies in the course of streaming and conduct postmortem analysis. We will describe the collaborative mechanisms, as well as the design principles and considerations in detail.

Index Terms: H.5.2 [Information Systems]: Information Interfaces and Presentation—User Interfaces; H.1.2 [Human information processing]: Collaboration—Event comparison

1 INTRODUCTION

The Mini Challenge 3 of VAST 2014 requires real-time visual analysis on streaming data. Given a streaming API, which pushed microblog messages and call center alarms continuously, we were asked to determine what was going on and to provide practical suggestions for authorities after the kidnapping of company employees. The major challenge lay in the "real-time" requirement, indicating that we had no time for specialized preprocessing and rudimentary exploration.

We decided to collaborate on analyzing the data, and thus proposed a highly flexible visual platform for assignment distribution and information sharing, employing automatic recognition techniques and allowing manual correction and synthesis. Meanwhile, for the keywords in the messages, we designed an interface to remind us of emergencies and events worthy of attention to assist in situation awareness.

A collaborative visual platform is efficient for analysts to synthesize information and share findings [2]. For streaming messages, the New York Times Team recently built a tool [1] to filter and organize the stream, which is extremely flexible and allows some co-oporations.

2 PLATFORM SPECIFICATION

2.1 Architecture

The architecture of the streaming message inspector is shown in Figure 1.

The data retriever performs preprocessing as soon as it receives each message from the remote server, including removing stop words, lemmatization and named entity recognition by courtesy of Standford NLP tools [1]. Then the processed data go into a database serving for a centered node: the coordinator.

*e-mail: {zipeng.liu, wangzhenhuang, csm, zuchao.wang, miaozhengjie, xiaoru.yuan}@pku.edu.cn

IEEE Symposium on Visual Analytics Science and Technology 2014
November 9-14, Paris, France
978-1-4799-6227-3/14/$31.00 ©2014 IEEE

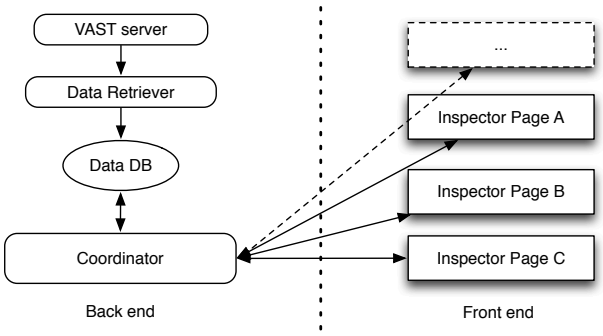

Figure 1: Platform pipeline. A centered server is in charge of message distribution and operative coordination.

Each member of the analysts team can open one or multiple inspector pages, which are connected together through the coordinator. The star topology is suitable for a small number of clients (inspector pages), due to its simplicity and elegance. Because we do not care about conflicts during the collaboration, consistency problems are not addressed. Therefore, the coordinator acts mainly as a message passer indeed: it broadcasts the information to all connected clients whenever it receives any imperatives or data messages.

2.2 Streaming message inspector

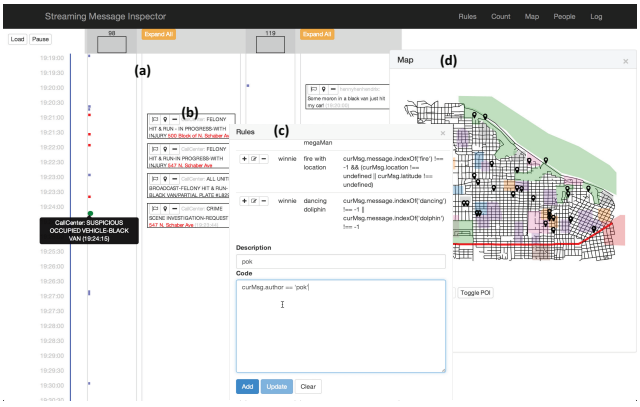

Figure 2: Streaming message inspector interface. (a) Substream; (b) message content; (c) rule panel; (d) map panel.

We divided the whole message stream into multiple substreams, according to filtering rules specified by users. Each message is represented as a small glyph along the timeline, and can be expanded fully if needed. Rules, locations and people names are manipulated in separate panels in the same page, as shown in Figure 2.

2.2.1 Rules

The filtering and organizing tasks are the most important ones when encountering massive amount of messages heterogeneous in content. The principles and rules of doing this will vary because of different scenarios and demands, such as eliminating advertisements, singling out messages mentioning a particular person, alerting upon located messages from call center. We employed one of the most flexible ways – writing javascript code snippets – to create filters, which removed the needs to design another new visual or non-visual language to translate the users' purposes, and dispelled potential obscure interactions. Yet, the shortcoming is also obvious: only a programmer knows how to manipulate the rules.

2.2.2 Map

The second important component of the inspector is the map, since locations provide direct information to answer the questions. We automatically pin an icon on the map whenever a message with location comes in, or highlight the mentioned streets (as many call center message came with two street names and sometimes even house numbers, and the intersection can be easily perceived). Moreover, we enable manual tagging on the map if the users know the exact place of the message, embracing human synthesis to support the analysis.

2.2.3 People Names

People are the crucial part in the answer. We set up a name recognition service using the Stanford Toolkit for automatic detection of people names. Besides, we can tag some words as people names by selecting the text directly, which will go into a names pool for later detection. If the same name is tagged manually before, the preprocessor will recognize it from that moment on.

2.2.4 Collaboration

To support close collaboration, and also prevent interference, we had to determine which piece of information should be shared and which should be kept privately. We regarded operations (create, update, delete, and etc.) of the rules, name tagging, locations as important ones, of which other members should be aware. For example, we designated one person, who knows javascript, to maintain a list of rules for others: the rules he created or updated were instantly reflected to other clients. One member pinned a message on the map, and then appropriate animations were displayed on the others' map panel.

During the final streaming phase, in which we are required to answer questions in near real time, we distributed the work to all the team members, with each one in charge of several rules and districts, monitoring and synthesizing the whole data stream together.

2.3 Keyword Analyzer

We designed the keyword analyzer to help users detect exigent events and extract key contents quickly. As shown in the Fig.3, this view consists of four components: keywords overview, overview timeline, single keyword timeline and data list.

In the keywords overview, high frequency keywords of all microblog records received so far are listed along with their frequency. Keywords in red mean they broke out at that time and deserve special attention.

The overview timeline displays all microblog messages. When a user clicks on a keyword, a timeline which corresponds to the keyword will appear in the single keyword timeline view. The user is able to brush on the timeline to set an time range. As the response, words which frequently co-occur in the selected time range will be listed below the timeline, as illustrated in Fig. 3 (e). Meanwhile, The data list view shows all the messages that include keywords appearing in the selected time range.

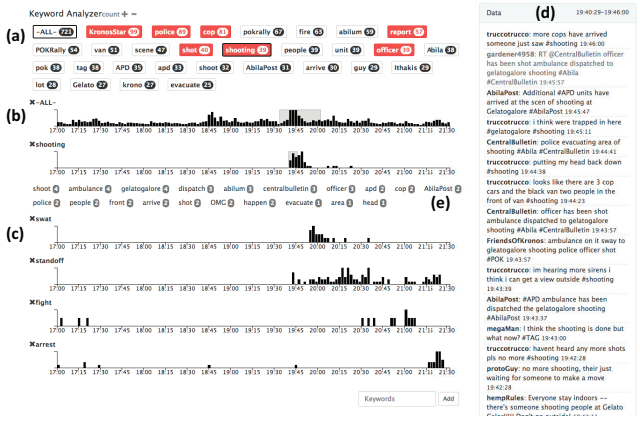

Figure 3: Keyword analyzer interface. (a) keywords overview; (b) overview timeline; (c) single keyword timeline; (d) data list.

3 Case Studies

In the following, we briefly discuss an event we found for Mini Challenge 3 to evaluate our platform.

Starting from the keyword analyzer, we are notified on emerging hot keywords, and then added the correspondent rules to monitor the situation. As shown in Fig. 3, this event began with a shooting, and then APD SWAT arrived at the scene. Standoff and negotiation between the shooters and the police nearly went through the whole period. At around 21:00, the two terrorists fighted with each other and were finally arrested.

By adding rules to filter the message stream, we were able to observe the event from different perspective, such as the hostages (@roger_roger and @prettyRain), and witnesses. Tracking objects that showed up throughout that time period like a black van, we discovered potential temporal-spacial relationships between several events happened in different locations of the city.

4 Discussion

This mini challenge was unprecedented in that it concerned streaming data. Although previous attempts in visualizing social media like Tweeter, Sina Weibo, were based on data streamed from an API, most experiments were conducted in retrospect. Seldom was deployed to real-time usage or made actual impact. Our platform is a bit austere and simple in functions since we address the flexibility, in order to deal with potential emergencies after the final streaming data segment began. Also, the effectiveness of it was highly dependent on the quality of filtering rules: if setting up bad rules, we might get huge noise (unwanted messages) or nothing. Thus, the platform requires several people to synthesize information and create effective rules for better usage. We will consider how to mitigate the pressure on the operator and team members by utilizing computer intelligence in the future.

Acknowledgements

The authors wish to thank IEEE VAST Challenge committee and reviewers. This work is supported by NSFC No.61170204.

References

[1] Streamtool, new york times rd lab. http://nytlabs.github.io/streamtools/.

[2] A. Wu, G. Convertino, C. Ganoe, J. M. Carroll, and X. L. Zhang. Supporting collaborative sense-making in emergency management through geo-visualization. *International Journal of Human-Computer Studies*, 71(1):4–23, 2013.