Find the Butterfly: A Social Media based Arterial Incidents Detection and Causality Analysis System

Kaiqun Fu[1], Weisheng Zhong[1], Chang-Tien Lu[1] and Arnold P. Boedihardjo[2]

[1]{fukaiqun, zwscn123, ctlu}@vt.edu

[2]arnold.p.boedihardjo@usace.army.mil

[1]Department of Computer Science, Virginia Polytechnic Institute and State University, Falls Church, VA, USA

[2]U. S. Army Corps of Engineers, Alexandria, VA, USA

ABSTRACT

Traditional statistical analysis on speed, volume, and occupancy has dominated the field of Arterial Incident Management Study (AIMS). However, few previous works have focused on investigating into the causality of the incidents. In this paper, we present *ButterFly*, a social media based arterial incident detection and analysis system. The proposed system is dedicated to identify the traffic incident from a novel perspective and discover causalities between traffic incidents. The main functionalities of the proposed system include: 1) Traffic incident detection based on user-input social media contents, 2) Transportation incidents storyline generation, and 3) Traffic incidents causalities analysis and visualization. We demonstrate the system by considering the Washington DC area as our experimental environment. *ButterFly* is targeted to provide effective and convenient real-time and historical traffic incidents analysis interfaces for transportation management agencies and academies. Our proposed system, integrated with multiple social media resources, can greatly broaden the visions for traffic incidents analysis.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS, Data mining*; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

social media, event detection, storyline generation, incident management

1. INTRODUCTION

Traffic incident early detection and historical traffic incidents analysis are two major problems in the field of transportation management. Current solutions to these two problems are dominated by the traditional methods for decades. For example, traffic incidents detection, are concurrently conducted by reports of roadway operation patrollers and CCTV (closed-circuit television) monitoring [1]. In most of the major cities, the CCTV system is the most important means to detect and verify traffic incidents. [3] However, incident detection delay or incapability may take place due to the low percentage of CCTV coverage. The historical traffic incident analysis, on the other hand, has been studied for a very long period of time [5]. However, such field of study is dominated by the means of traditional statistical analysis methods focusing on magnetic loops, weather or environment data [4]. Such methods are effective, but contemporarily insufficient. With recent advances in data collection, sensors technologies, and social media networks, the problems of traffic incident early detection and historical analysis can be resolved in a perspective of a new entry point. In the past decade, as social media (e.g., Twitter, Facebook and Waze) became more popular [2] , social media data has been collected and used in various applications.

Inspired by the aforementioned problems and also motivated by the age of social media, we developed the *ButterFly* system, a user friendly mobile application that provides efficient traffic incident detection functionality. A real-time geocoding process is later applied for retrieving the geographical coordinates. Furthermore, the traffic incident storyline generation provided by *ButterFly* focuses on summarizing and rearranging the contents on social media networks, according to the evolution of the traffic incident. These functionalities will allow the transportation agencies to gain a supplementary view of the traffic incidents overtime. The major contributions of can be summarized as follows:

- **Proposing a social media based traffic incidents detection platform**: The system applies query expansion techniques to collect transportation related tweets from Twitter. An efficient data transmission scheme is established to maintain the large amount of transportation related tweets.

- **Developing a storyline generation algorithm for transportation events**: A state-of-the-art nature language processing strategy is designed for identifying

SIGSPATIAL'15, November 03 - 06 2015, Bellevue, WA, USA

http://dx.doi.org/10.1145/2820783.2820797.

the names of the road sections in short textual contents, for example, the tweets content.

- **Designing a traffic incidents causalities analysis tool**: The proposed system addresses and presents a novel perspective towards the traffic incidents analysis. Few previous works focus on the visualization of the causalities and storytelling between traffic incidents.

- **Visualizing the incidents detection and storyline generation**: A user friendly web-based application interface was developed to combine all the above functions. It utilizes several state-of-the-art web technologies to provide an efficient and convenient user experience.

2. SYSTEM ARCHITECTURE

This section describes the system architecture for *ButterFly*. At the higher level, the system consists of three main components: data pre-processing and storage, application services, and the user interface.

2.1 Application Services

Two types of databases are maintained in the backend of *ButterFly*: the relational database consists of a traditional relational PostGreSQL with PostGIS database for spatial Waze traffic datasets and shape files for Metropolitan Washington DC area, while the NoSQL database stores Twitter data for text mining on social media. The main web service is implemented Python Flask environment, due to its highly collaborative with the two deployed databases, enables us to move on to further processes more smoothly.

2.1.1 Street Entities Extraction

Streets entities extraction is one of the major preprocessing steps of the proposed system. Motivation of implementing this module is to extract the mentioned street names out of the transportation related tweets. In order to fulfill this target, we applied the *AlchemyAPI*[1]. It uses advanced deep learning methods to solve many problems in natural language processing. For example, from our Twitter database, the tweet "*I-395 S near VA-120/Glebe Rd (Ex7), a crash blocks the right lane. #VATraffic #DCTraffic.*" is one transportation related tweet posed by the influential user "*WTOPTraffic*". One can manually identify three streets entities: *I-395 S*, *VA-120*, and *Glebe Rd* there are all arterials roadways in the Metropolitan Washington D. C. area. The output of our Street Entities Extraction module, our system can obtain the aforementioned street names systematically. Part of the results are shown in Table 1. In which all bold font words are extracted street entities.

Table 1: Street Entities Extractions from Tweets

I-395 S near **VA-120/Glebe Rd** (Ex7), a crash blocks the right lane. #VATraffic #DCTraffic
In **Old Town Alexandria**, **lower King St** and portions of **Union Street** both ways reported flooded. #VATraffic #DCTraffic @WTOP
US-1 near **VA-286/Fairfax Co Pkwy** is closed both ways due to flooding #VATraffic #DCTraffic

[1] *http://www.alchemyapi.com*

2.1.2 Geocoding Process

The *Geocoding* module takes the outputs from the *Streets Entities Extraction* module. The purpose of implementing this module is to pinpoint the extracted street names on a map. To accomplish this task, coordinates of the streets entities are required. In order to obtain the coordinates, a *Geocoding* module is necessarily needed. In our case, we utilized the *Google Geocoding APIs*[2]. For instance, the extracted street entities are: *I-395 S*, *VA-120*, and *Glebe Rd*, the proposed system is capable of retrieving the coordinate (*38.8467565, -77.0781714*), corresponding to the mentioned entities.

2.2 System Interface

User interactions and operations such as sending requests and receiving results or feedback sent back from the backend server are performed by this module. This web-based mobile application utilizes Sencha Touch to construct the framework. All communications to the backend Python Flask web server were implemented by Ajax technology. Google Maps APIs were leveraged to enable location based services such as traffic incident detection display, street entities identification geocoding, and historical traffic incidents storyline generation.

3. FEATURES

ButterFly is capable of crawling and extracting transportation related tweets from the Twitter server. The dynamic crawling process developed can be applied to multiple cities. The street entities extraction and geocoding module and the historical traffic incidents storylines generation module are the most important feature of the system. The entities extraction module learns the patterns of the transportation related tweets, extracts the location entities out of short textual contents like tweets, and implements geocoding service to retrieve the geographical coordinates of the mentioned location. On the other hand, the storyline generation module summarizes the tweets reporting the same traffic incident and constructs a highly comprehensible storyline based on the evolution of the traffic incident.

3.1 Traffic Incidents Detection

A data collector utilizes the Twitter timeline API to retrieve the latest tweets by a set of predefined influential users under the transportation topic. Four influential users were initially selected by the system, namely, "*WTOPtraffic*", "*VaDOT*", "*drgridlock*" and "*DCPoliceDep*". These users are active under the topic of transportation in the Washington DC area. Figure 1 illustrates the process of data collection and tweet query expansion.

A collection of initial transportation related tweets set T_I was extracted from the influential users, after which a *tf-idf* ranker was applied to the set of extracted tweets T_I, defining words with high *tf-idf* scores as keywords in list Q because we are confident that the words that frequently occur in their tweet content are more likely to be transportation related. Then, this new keywords list Q became the expanded query to be inputted into the *Twitter Search API* to crawl all the tweets that match the queries.

The *Apriori* algorithm was leveraged to determine the association rules between the words in T_I. The reason we chose

[2] *https://developers.google.com/maps/documentation/*

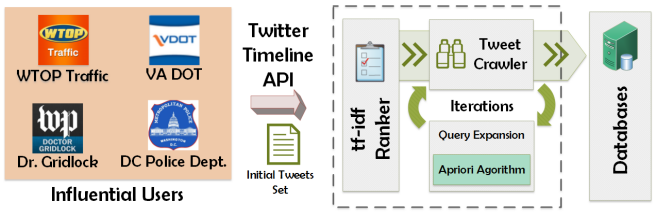
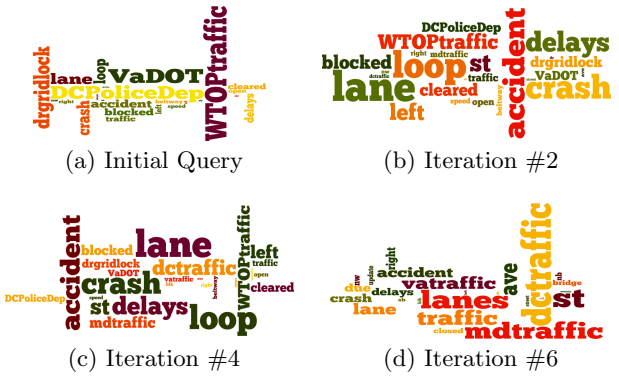


Figure 1: Data Collection and Tweets Query Expansion

to utilize association rules algorithms was to avoid generating a long query with single words. If not applied, our data crawler would only send one word at a time as the query for the *Twitter Search API*, but by applying this algorithm any one of "crash", "right", "lane" entered separately as queries will retrieve all tweets containing the word "crash". There are several obvious problems with this method, however: not only would the noise in these results be enormous, the high number of web requests sent means that the data crawler will rapidly reach its rate limitation.

In order to solve these shortcomings, the *Apriori* algorithm was applied to identify the minimum support wordsets in the vocabulary lattice. The concept of "wordsets" is equivalent to "itemsets" from items transaction mining. One of the minimal support wordsets serves as the new query, e.g., ["*crash*", "*lane*"], and logic connections provided by *Twitter Search APIs* join these two or more words together as a more complicated query: "*crash AND lane*".



(a) Initial Query (b) Iteration #2

(c) Iteration #4 (d) Iteration #6

Figure 2: Word Cloud for Query Expansion

Figure 3 shows the word clouds for the initial query and resulting queries after iterations of expansions. Figure 2(a) shows the word cloud for the original query generated by the influential users' tweets sets T_I. Figures 2(b), 2(c), and 2(d) show the queries generated by iterations two, four and six, respectively. These results indicate that the query is changing gradually from a specific topic (set of keywords) that focuses solely on the influential users to a more general transportation related topic. Based on our observations, the query will eventually converge after an average of 6 query expansion iterations has been performed. The expanded query broadens the searching space while maintaining appropriate filtering keywords. It thus helps to retrieve more traffic related data while preserving the data quality.

3.2 Incident Storyline Generation

The incident storylines generation module is capable of sorting and reasoning the traffic incidents that extracted by the traffic incident detection module. In the research field of transportation incidents management, there is little existing work for traffic incidents storyline generation. In the proposed system, storylines are constructed for the purpose of monitoring and analyzing the evolution of the traffic incidents. A spatial-temporal-textual similarity calculation algorithm is utilized as metric for generating the story lines.

In order to calculate the similarities between the tweets, the traffic incidents related tweets are all vectorized by the extracted entities. The entities are the outputs from the street name entities extraction module in the application service.

$$\mathbf{Twt} = (E_1, E_1, ..., E_N)^T \tag{1}$$

where E_i represents the occurrence of the i^th entity in one tweet, $i \in N$, here the tweet is represented in the format of vector: $\mathbf{Twt}$. For example, tweet: "*US-1 near VA-286/Fairfax Co Pkwy is closed both ways due to flooding #VATraffic #DCTraffic*" will be mapped to a vector with ones at positions at entities "*US-1*", "*VA-286*", and "*Fairfax Co Pkwy*"; zeros at positions otherwise.

In the proposed system, cosine similarities between the tweets are calculated, given the tweets vectors. In addition, a time window with appropriate size is defined to filter out the temporally irrelevant tweets. The process is represented as:

$$\forall i \neq j, Sim(\mathbf{Twt}_i, \mathbf{Twt}_j) = \frac{\mathbf{Twt}_i \cdot \mathbf{Twt}_j}{|\mathbf{Twt}_i| \times |\mathbf{Twt}_j|} \tag{2}$$

where $\mathbf{Twt}_i, \mathbf{Twt}_j \in T$. T is the time window, in the proposed system, the window size is predefined as 4 hours. If the Sim between $\mathbf{Twt}_i$ and $\mathbf{Twt}_j$ is greater than a threshold, we consider $\mathbf{Twt}_i$ and $\mathbf{Twt}_j$ are saliently similar. In other words, these two tweets are highly likely to be referring same traffic incident. In the proposed system, we predefine the similarity threshold as 0.4. Figure 3 shows one storyline generated by our algorithm.

4. DEMONSTRATION SCENARIOS

ButterFly user interface is designed and demonstrated using a real word database. The cases studies show that *ButterFly* is very useful and practical under certain scenarios.

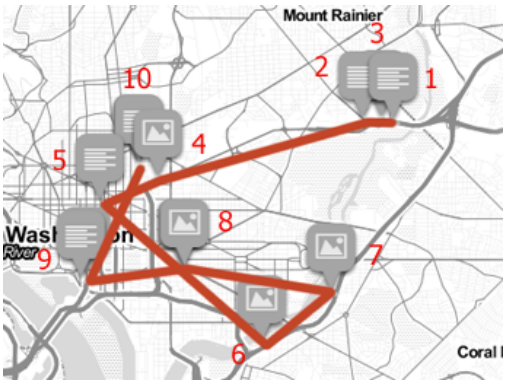


Figure 3: Story Map for May 12th 2015

4.1 Scenario 1. Incident Detection

The main user interface is shown in Figure 3, which allows users to search for historical incidents based on user specified requests. And an incident chain sorted by the time stamp will be generated displaying all incidents detected in the day. This feature is designed for a more straightforward understanding to the causalities between incidents. For one particular incident, several types of information are presented: 1) Media description: twitter images or video of incident 2) Time stamp: the detected incident occurrence time. 3) Location: the coordinates and street name where the incident happened. 4) Timeline: the story we generated from Twitter Dataset that related to the incident. For example, in Figure 3, an incident chain is shown for all the incidents of *May 12th 2015*. Markers represent incidents' locations, while the numbers show the temporal sequence of the incidents.

Spatio-temporal information, incident story (Timeline, details described in Scenario 2), and media description will be shown under the same interface. For example, Figure 4 shows an incident took place at *New York Ave NW, May 12th 2015*. Spatial information (New York Ave NW), temporal information (*May 12th 2015 9:25 AM*), an incident story, and media description are displayed. By clicking on the next or previous icon, or any marker on the incident storymap, the interface will focus on the selected incident.

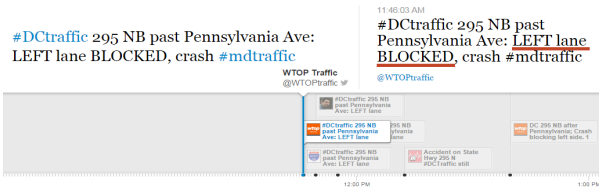
Figure 4: Single Incidents

4.2 Scenario 2. Incidents Story Telling

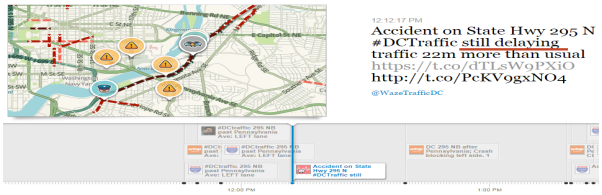
In order to show the evolution of the traffic incidents, traffic incident storylines are generated. Tweets that related to the incident are collected and used to construct an incident storyline sorted by time stamp. Figure 5 shows the evolution of an incident happened at *State Hwy Interstate 295 N, May 12th 2015*: Figure 5(a) shows the first incident reporting tweet posted by the user *WTOPTraffic* at *11:46 AM*. It mentions that vehicle crash occurred on the left lane of interstate highway 295 north bond. About 20 minutes later, around *12:12 PM*, another tweet is posted by the user *WazeTrafficDC*, as shown in Firgure 5(b). It updates the status of the detected incident, stating that the traffic is still delaying. Figure 5(c) shows a tweet, posted by the user *WTOPTraffic*, informs that the crash at interstate 295 north bond is cleared at *1:18 PM*. This generated storyline explicitly indicates the evolution of the detected traffic incident from its appearance to its clearance.

5. CONCLUSIONS

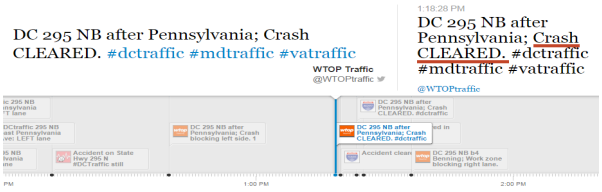
ButterFly is designed to meet the needs of an end user who wishes to find the causalities and evolutions between traffic incidents. The transportation event detection feature helps the transportation incident managers to make faster and more precise decisions. *ButterFly* implements traffic incidents storylines generations to monitor the evolution of the



(a) First Incident Reporting Tweet



(b) Incident Status Updating Tweet



(c) Final Incident Clearance Tweet

Figure 5: Traffic Incident Evolution Case

traffic incidents. This feature visualizes the traffic incidents' status. *ButterFly* is envisioned to eventually go beyond city level coverage and evolve into a collaboration platform that functions nationwide to improve the overall quality of traffic incidents analysis

6. ACKNOWLEDGMENT

This work is supported in part by the District of Columbia Department of Transportation (DCDOT) under contract number DCKA-2015-C-0029. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DCDOT or the DC Government.

7. REFERENCES

[1] N. Buch, S. Velastin, J. Orwell, et al. A review of computer vision techniques for the analysis of urban traffic. *Intelligent Transportation Systems, IEEE Transactions on*, 12(3):920–939, 2011.

[2] K. Fu, Y.-C. Lu, and C.-T. Lu. Treads: a safe route recommender using social media mining and text summarization. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 557–560. ACM, 2014.

[3] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Traffic monitoring and accident detection at intersections. *Intelligent Transportation Systems, IEEE Transactions on*, 1(2):108–118, 2000.

[4] P. Songchitruksa and K. Balke. Assessing weather, environment, and loop data for real-time freeway incident prediction. *Transportation Research Record: Journal of the Transportation Research Board*, (1959):105–113, 2006.

[5] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.