Steds: Social Media based Transportation Event Detection with Text Summarization

Kaiqun Fu
Virginia Tech
Falls Church, Virginia, 22043
Email: fukaiqun@vt.edu

Chang-Tien Lu
Virginia Tech
Falls Church, Virginia, 22043
Email: ctlu@vt.edu

Rakesh Nune
District of Columbia
Department of Transportation
Washington DC, 20003
Email: rakesh.nune@dc.gov

Jason X. Tao
District of Columbia
Department of Transportation
Washington DC, 20003
Email: jason.tao@dc.gov

***Abstract*—Ubiquitous user-input contents on social media and online services have generated a tremendous amount of information. Such information has great potential applications in various areas such as events detection and text summarization. In this paper, a social media based traffic status monitoring system is established. The system is initiated by a transportation related keyword generation process. Then an association rules based iterative query expansion algorithm is applied to extract real time transportation related tweets for incident management purpose. We also confirm the feasibility of summarizing the redundant tweets to generate concise and comprehensible textual contents. Comparison results show that our query expansion method for tweets extraction outperforms the previous ones. Analysis and case studies further demonstrate the practical usefulness of our tweets summarization algorithm.**

***Index Terms*—component, social media analysis, data mining, intelligent transportation systems, tweets extraction.**

I. Introduction

Early detection of traffic incidents is critical to reduce the impact of incidents on the traffic conditions. There are multiple incident detection sources which include reports of roadway operation patrollers, citizen calls, CCTV (closed-circuit television) monitoring and alerts from vehicle detection stations. In most of the major cities, the CCTV system is the most important means to detect and verify traffic incidents. However, incident detection delay or incapability may take place due to the low percentage of CCTV coverage. These drawbacks motivate us to explore the application of social media data analysis in incident detection and traffic management.

In the past decade, as social media (e.g., Twitter, Instagram and Facebook) became more and more popular, social media data has been collected and used in various applications. For instance, Xie et al. [1] explored the social media data to predict and track disease outbreaks. Jafarzadeh [2] explored the benefits of integrating social media tools into emergency communications and monitoring the social media contents during an emergency or disaster. Schulz et al. [3] developed an approach for efficient processing and storing of social media data for emergency management purpose. Applying social media analysis in traffic management is a relatively new concept. Cui et al. [4] proposed a prototype system that can capture and publish traffic status generated by a Chinese social network (Sina Weibo). Today, many government organizations and news companies are using social media such as Twitter to communicate with the public and commuters on traffic incidents. On the other hand, many commuters report incidents, special events or congestion levels that on route through the social media. Therefore, the contents of popular social media contain abundant information on traffic incidents and roadway congestion. It is feasible to extract and analyze the social media data for detecting traffic incidents and obtaining supplemental incident information.

However, the abundant information provided by Twitter is sometimes redundant; this is due to two factors: 1) Twitter retweets: To repost or forward another user's message on the social networking website Twitter; 2) Different eyewitnesses can post tweets with every identical contents about the same incident. These types of information redundancies can be an obstacle for the incident management systems which consider social media as their building blocks. For instance, important incidents related tweets can be buried under a stack of tweets discussing the same trivial incident. [5] In order to prevent the needles from being buried in the haystack, text mining and natural language processing techniques [6] [7] are used to summarize the redundant tweets with similar meanings. For instance, Fu et al. [8] proposed a summarization system for social media contents such as Yelp restaurant reviews and tweets to enhance the routing performance.

In this paper, we present *Steds* (*Social Media based Transportation Event Detection with Text Summarization*), an efficient approach to extract and analyze real-time traffic related twitter data for incident management purpose. *Steds* takes the advantages of the sufficient information sources from the social media by applying the tweets extraction technique. On the other hand, it resolves the information redundancy problem by utilizing the tweets summarization algorithm. The major contributions of *Steds* can be summarized as follows:

- **Developing a real-time social media based traffic incidents detection platform**: By applying the client-server model, the proposed web based incident detection platform is capable of monitoring traffic related tweets in a local area. Moreover, such platform can be implemented to other regions.
- **Proposing a query expansion algorithm for traffic related tweets**: The proposed system applies query ex-

pansion techniques to collect transportation related tweets from Twitter server. A backend database is established to maintain the large amount of traffic related tweets.

- **Designing a text summarization method for redundant traffic related tweets**: extractive and centroid based summarization algorithm is implemented to summarize the redundant tweets, for the purpose of deducting similar contents.
- **Conducting extensive experiments for tweets extraction performance**: the effectiveness and efficiency of our tweets were evaluated on the convergence speed with existing approaches. The convergence speed is represented by two metrics.

The remainder of this paper is organized as follows. Section II presents technical details for the transportation related tweets extraction and query expansion algorithm. This section also introduces the redundant tweets summarization algorithm. Experiments and results on query expansion and tweets summarization are presented in section III. Conclusions are drawn in section IV.

II. Methodology

This section describes the detailed techniques and algorithms involved in *Steds*. An iteratively processed query expansion is designed on the fundamentals of an influential users set selection scheme. Real time tweet crawling procedure is performed for transportation related tweet extraction. An extractive summarization algorithm is designed to eliminate the tweets redundancy.

A. Keywords Generation

In order to successfully extract traffic incident related tweets, it is necessary to establish a set of keywords that will be sent over twitter to query data through the twitter timeline API [9]. We have identified four influential Twitter users (seed users) who actively post traffic information. These four account names are "*WTOPtraffic*", "*VaDOT*", "*drgridlock*" and "*DCPoliceDep*", which are held by *WTOP* (a popular radio station in metropolitan Washington DC area), Virginia Department of Transportation, the Washington Post and District of Columbia Police Department, respectively. Note that an assumption is made that we consider all tweets from these influential users are transportation related tweets. A data collector is built to collect 3200 recent tweets posted by four influential users that compose a document set, denoted as T_i.

A statistic method called "term frequency inverse document frequency" (or simply as *tf-idf*) is applied to document set T_i to determine the importance of each word in the set. *tf-idf* the product of two statistics, term frequency and inverse document frequency. For word t and document d, the term frequency *tf(t,d)* is defined as:

$$tf(t,d) = \frac{f(t,d)}{MAX\{f(w,d) : w \in d\}} \quad (1)$$

where *f(t, d)* is the number of times that term t occurs in document d.

The inverse document frequency, or *idf*, is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. *idf* is defined as:

$$idf(t,D) = log(\frac{N}{|\{d \in D : t \in d\}|}) \quad (2)$$

where D is a corpus of all document, N is the number of documents in the corpus, and $|d \in D : t \in d|$ is the number of documents where word t appears. In our case, each tweet is a document, and all tweets together compose a corpus. Since all words to be analyzed are from the extracted tweets, $|d \in D : t \in d|$ will never be zero. We choose a total of 50 words with highest *tf-idf* weights as the traffic incident related keywords.

Q_{Init} denotes the initial set of keywords extracted from the influential users. The keywords list Q_{Init} can be used as input to the Twitter Search API to crawl all the tweets that match the queries, and denote this set of tweets as T_1. A new query Q_i will be generated by the query expansion algorithms based on T_i from the previous day. The final keywords after iterations are shown in TABLE I. The data flow of the data collection and tweets query expansion algorithms are presented in Fig. 1

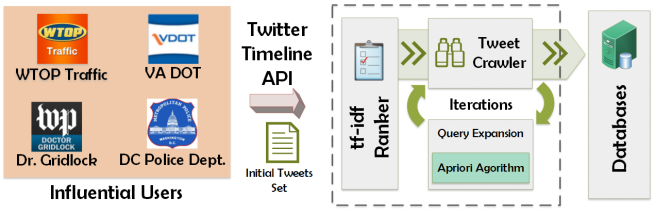


Fig. 1. Data Collection and Tweets Query Expansion

B. Query Expansion

Enormous noises in extracted tweets will be introduced when single keywords such as "*crash*" or "*lane*" is used as queries. In addition, the number of tweets from the data crawler will easily reach the assigned rate limit. The previous works proposed an *n-gram* based approach to solve the problems [4]. However, such method has significant drawbacks: 1) the parameter n in *n-gram* must be pre-determined; 2) *n-gram* is only capable of proceeding fixed number of grams, it cannot identify arbitrary length of keywords sets. In order to solve these issues, we applied the *Apriori* algorithm [10] to tweets set T_i to develop the association rules of keywords and then construct Twitter queries using a combination of a few keywords (or called "wordset") rather than single keyword.

The *Apriori* algorithm is usually applied in the field of transaction mining to establish frequent itemsets for boolean association rules. The algorithm proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. In general, the concept of the *Apriori* algorithm can be expressed in Algorithm 1.

where T denotes the total tweets for the previous time slot, in our system, we choose one day as the time interval. ϵ denotes the minimum support level. And the support level

TABLE I
TOP 50 KEYWORDS AFTER ITERATIONS

Word Freqency Rank	Keywords									
1-10	#dctraffic	#mdtraffic	#vatraffic	crash	due	delays	st	traffic	right	ave
11-20	lane	lanes	rd	nw	accident	buses	left	md	nb	bridge
21-30	sb	earlier	street	near	blocked	loop	congestion	expect	va	dc
31-40	update	road	work	following	closed	open	vehicle	inner	car	killed
41-50	new	get	minute	directions	close	schedule	police	beltway	operating	us

Algorithm 1: *Apriori* Algorithm for keywords (queries) generation

Function *Apriori*(T, ϵ)
  $L_1 \leftarrow \{large1 - keywordsets\}$;
  $k \leftarrow 2$; /* Set Initial Itemset Size to 2 */
  while $L_1 \neq \emptyset$ **do**
    $C_k \leftarrow \{a \cup \{b\} | a \in L_{k-1} \wedge b \in \bigcup_K L_{k-1} \wedge b \notin a\}$;
    for $Tweetst \in T$ **do**
      $C_k \leftarrow \{c | c \in C_k \wedge c \subseteq t\}$;
      for $keywordsc \in C_t$ **do**
        $count[c] \leftarrow count[c] + 1$; /* Update Frequency by 1 */
    $L_k \leftarrow \{c | c \in C_k \wedge count[c] \geq \epsilon\}$;
    $k \leftarrow k + 1$; /* Update Itemset Size by 1 */
  return $\bigcup_K L_k$

indicates the frequency of the word set in T. The support level for a given word set $L = \{word_1, word_2, ...\}$ is calculated:

$$Supp(L) = \frac{|\{a | c \in T \wedge L \subseteq a\}|}{|T|} \tag{3}$$

where $|T|$ denotes the total number of tweets is the target tweets set; $|\{a|c \in T \wedge L \subseteq a\}|$ indicates the number of supersets of L in T.

In general, *Apriori* algorithm can be summarized by following steps: 1) Identity frequent itemsets that are the sets of item with minimum support (noted by in the algorithm); 2) Apply the *Apriori* property: Any subset of frequent itemset must also be frequent. That means all subset of frequent itemset must have minimum support; 3) Extend the length of itemset by the join operation: To find L_{k+1}, a set of candidate $k+1$ itemsets is generated by joining L_k with itself.

In this case, each keyword is viewed as an item and thus a wordset is viewed as an itemset in the field of transaction mining. The minimum support level is set to 0.02 in this implementation. In order to highlight the most relevant tweets from the extracted dataset, all tweets are ranked based on the *tf-idf* weights of the contained keywords. For each tweet, a score is calculated by adding the *tf-idf* weights of all keywords appearing in the tweet. All extracted tweets are ranked by their scores and stored into the backend database.

Fig. 2 illustrates a simple example, showing how the *Apriori* algorithm can be utilized to identify the keywords sets. For example, after stop words removal being processed towards T, four single words are left with actual meanings: "*crash*", "*lane*", "*right*", and "*left*". A word lattice illustrated in Fig. 2 is then constructed for all possible combinations of the single words. Each node in Fig. 2 represents a word set L_i. After applying the *Apriori* algorithm, the lattice is left with all L_is with $Supp(L_i) > \epsilon$, corresponds to the nodes above the red curve cut in Fig. 2. Then the *maximal frequent word set* is identified for this lattice, denoted by the red circles in Fig. 2. Note that: a *maximal frequent set* is a frequent set for which none of its immediate supersets are frequent.

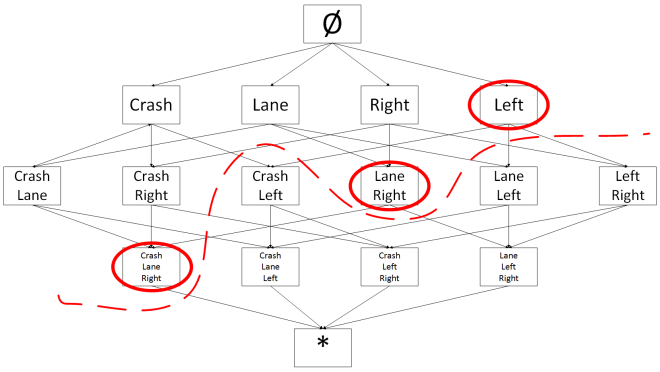


Fig. 2. Wordset Lattice for Apriori Algorithm

C. Tweets Summarization

The motivation for implementing text summarization techniques is to reduce and eliminate the need to read multiple similar texts by replacing the individual objects with a summary of the transportation topic related tweets. The technique involved fall in the scope of the extractive summarization algorithms. Based on these algorithms, sentences in each document are represented by nodes in a complete graph, and the cosine similarities are represented as the edges between the nodes. Those edges with a cosine similarity less than a predefined threshold will be removed. After the edge removal process, summarization algorithms based on *PageRank* or *LexRank* is applied to this graph. The proposed system applies *LexRank* based algorithms, since the *PageRank* based algorithms are more generally used for directed graphs. The topic ranked sentences are considered as the summary for the documents.

For the purpose of determining the correlation between tweets and identifying the most salient one out of the set of tweets with similar content, the similarities between the tweets are quantified with the *idf-modified-cosine* similarity, given by:

$$idf - modified - cosine(x, y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}} \tag{4}$$

where the term $tf_{w,t}$ is the number of occurrences of the word w in the tweet t. The *idf-modified-cosine similarity*, in our scenario, can also be considered as the $salience$ of the tweet. Fig. 3 shows an example of weighted *idf-cosine similarity* graph. The nodes in the graph are the tweets and the edges in the graph are the *idf-modified-cosine similarity* between two tweets. The weight of the edge correspond the similarity value of between two tweets. Note that we classify the value into three levels: $similarity \in [0.0, 0.6]$, $similarity \in [0.6, 0.8]$ and $similarity \in [0.8, 1.0]$.

In order to identify the most salient tweet among the similar tweets set, the edges with low salience ($similarity \in [0.0, 0.6]$, the dash lines) are removed from the similarity graph. This removal can be intuitively considered as graph separation, the result would be several separated graph components. For example, the solid edges and the nodes in Fig. 3 construct one component. Each tweet in one component has high similarity between all other tweets in the same component, and low similarity with tweets outside the component.

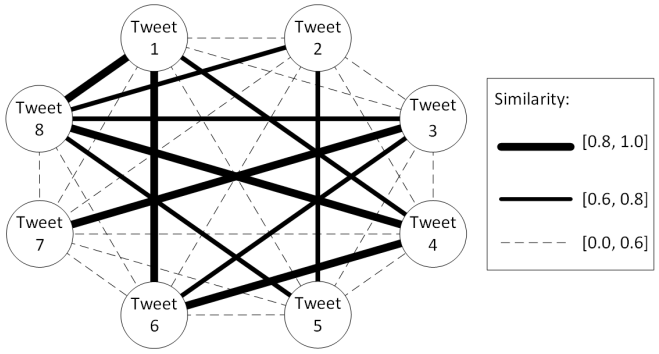


Fig. 3. Weighted Cosine Similarity Graph

Within one specific component, a *LexRank* score can be calculated by applying the following methods:

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{deg(v)} \tag{5}$$

where $p(u)$ is the centrality of the node u, $adj[u]$ is the set of nodes that are adjacent to u, and $deg(v)$ is the degree of the node v. Equation (5) can be written in the matrix notation as:

$$\mathbf{p} = \mathbf{B}^T \mathbf{p} \tag{6}$$

where the matrix $\mathbf{B}$ is obtained from the adjacency matrix of the similarity graph by dividing each element by the corresponding row sum. As illustrated in equation (6), $\mathbf{p}^T$ is the left eigenvector of the matrix $\mathbf{B}$ with corresponding eigenvalue of 1. Then the *LexRank* score can be calculated recursively as $\mathbf{p}^T$ is given.

III. Experiments and Analysis

In this section, the performance of the proposed algorithm is evaluated. In the experiments, we introduce two metrics and two comparison methods that are proposed previously in the literature. Case studies based on real data are presented for demonstrating the practical applications of the proposed summarization algorithm.

A. Experiment Data

We utilized Twitter timeline API [11] for collecting the historical data of the influential users. Search API was processed for crawling the real time Twitter data. Parameters for this API includes a query, which we iteratively generate from the query expansion algorithm; as we consider Metropolitan Washington D. C. area as our experimental environment, a geocode parameter is set to be the center of Washington D. C., with a radius of 10 miles: "*38.897345, -77.036196, 15mi*". Data crawling process has been started since September 2013, and it has been processed 24 hours a day, 7 days a week. The overall dataset has a time span of 20 months; a size of 28.6 GB. All datasets are stored in MongoDB, server is provided by the District of Columbia Department of Transportation.

B. Query Expansion Experiments and Results

For the purposes of empirical study, the evaluation of the query expansion algorithm plays an important role in proposing the algorithm. However, our data source and datasets have several special properties: 1) our Twitter data source is real time, 2) the datasets are contents of tweets, very different from the target datasets such as articles or web pages that traditional query expansion algorithms focus on, and 3) lack of ground truth data for results comparison. Due to these properties, it is hard to evaluate the precision of our generated queries. Nevertheless, according to our observations to the iteratively generated queries, a pattern of convergence was found. The evaluations can be processed in an alternative perspective, namely, the speed of the query convergence. The intuition of such method is to calculate the difference between two consecutive queries, a convergence would be observed if the difference is converging to zero. In later parts of the subsection, two metrics will be introduced to be indicators of the rate of convergence, and two pervious methods will be compared against our algorithm.

1) Metrics: Two metrics were adopted to evaluate the results of all the methods tested:

- **Cosine Similarity**: As a well-recognized metric for measuring the text proximity, cosine similarity is used to evaluate the similarity between two consecutive queries:

$$cos(\vec{Q_d}, \vec{Q_{d-1}}) = \frac{\vec{Q_d} \cdot \vec{Q_{d-1}}}{|\vec{Q_d}| \cdot |\vec{Q_{d-1}}|} \tag{7}$$

  where the term $\vec{Q_d}$is the vector formalized for the query at time interval d. In our system, the time is one day. A valid queries comparison is made between day d and day $d-1$.
- **Jaccard Index**: A statistic way comparing the similarity and diversity of sample sets. In our system, the queries are considered as sets of words, it is capable of defining

the similarity of two consecutive queries in a different perspective:

$$J(Q_d, Q_{d-1}) = \frac{Q_d \cap Q_{d-1}}{Q_d \cup Q_{d-1}} \quad (8)$$

where the term Q_d is a set presentation of the query Q at time interval d. And every word in the query represents the element in the set.

2) Comparison Methods: Our proposed query expansion algorithm is compared with two existing methods.

Plain tf-idf model: This method is based on the ranking of the *tf-idf* score. And this model is solely dependent on the *tf-idf* score. *Plain tf-idf* models detailed strategy is: 1) separate the previous days tweets by 1 hour time interval consider each time interval as a document; 2) calculate the *tf-idf* score for each unique word (vocabulary) in the previous days tweets; 3) rank the words based on their *tf-idf* scores, and select the top 50 words as the query for the next day. Note that all words contained in queries generated by this model are single-words.

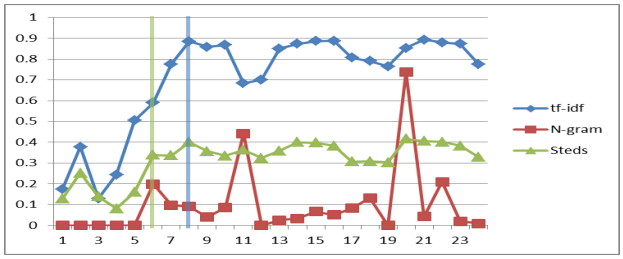
N-gram model: This model was proposed by Cui et al. [4] for extracting traffic information from social media interactions. The detailed strategy is: 1) define the parameter N for the number of words contained in the word set query; 2) calculate the occurrences for each N-word set; 3) rank the N-word sets based on their occurrences in all tweets from the previous day. Note that the words contained in queries generated by this model have a fixed length of N.

3) Performance Analysis: Fig. 4(a) shows the results for convergence speed under different metrics. In Fig. 4(a) we cannot find a clear pattern of convergence for the *N-gram* model; as for the *tf-idf* model, the *cosine similarity* converges with a relatively slow speed (at day 8); we find that our proposed *Apriori* based query expansion algorithm (*Steds*) achieves the fastest query convergence speed (at day 6). In Fig. 4(b), similar patterns can be found from the perspective of *Jaccard Index*, however, the convergence speeds reflected by *tf-idf* model and *Apriori* based model do not show a significant difference.

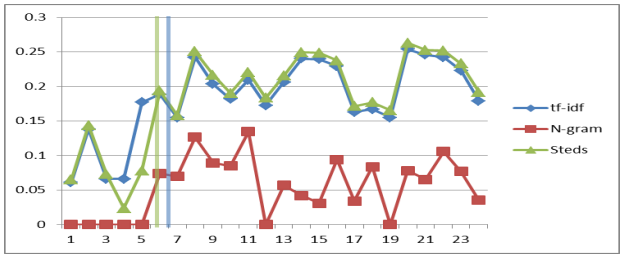
Note that according to observations, two interesting facts are observed: 1) **Event Sensitivity**: some special or unexpected events occurrence can affect the convergence of the query locally, as shown in Fig. 4(a), a local peak at the end of the curve; 2) **Weekly Periodicity**: after the queries have converged, a weekly fluctuation is presented: relatively high cosine similarities on weekdays, weekends have low cosine similarities. This pattern shows that on weekdays, the transportation related tweets share more stable keywords. On the weekends, however, the shared keywords vary more intensively.

4) Results Analysis: A daily tweets volume distribution analysis is studied, from which a weekly pattern is discovered. Also, the evolution of the queries is demonstrated in this section.

Daily Tweets Volume Distribution: An intuitive user interface is built to display the latest traffic tweets along with visualizations tools like histogram in *Steds*. The implemented



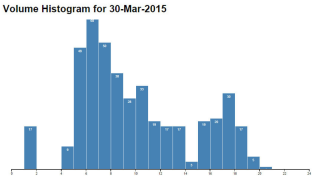
(a) Cosine Similarity



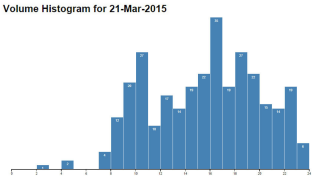
(b) Jaccard Index

Fig. 4. Performance Evaluations with Cosine Similarity and Jaccard Index

user interface allows the users to search for the incident or congestion related tweets on particular dates and times. This feature helps the operators at traffic management center to obtain supplemental incident information which may be missing from the existing incident profiles. Tweets volume histograms for a regular work day and a weekend day are displayed in Fig.5(a) and 5(b). The figures clearly shows that in a regular work day, more traffic related tweets are posted during morning and afternoon rush hours. In a weekend day, most traffic related tweets are posted in the evening time from 5:00PM to 8:00PM. This is consistent with the travel rates and incident rates over daily time.



(a) Work days Tweets Volume

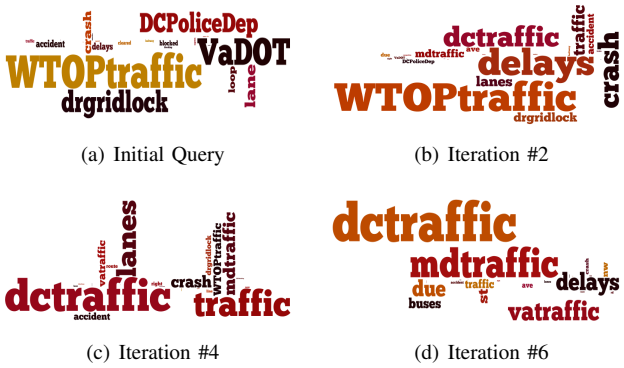


(b) Weekends Tweets Volume

Fig. 5. Tweets Volume Distributions on Different Dates

Evolution of Query Expansion: Fig. 6 shows the word clouds for the initial query and resulting queries after iterations of expansions. Fig. 6(a) shows the word cloud for the original query generated by the influential users tweets sets T_i. Fig. 6(b), 6(c), and 6(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.



(a) Initial Query (b) Iteration #2

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

Fig. 6. Word Cloud for Query Expansion

C. Case Studies for Summarization Algorithm

Case studies are provided in this section for demonstrate the practical usefulness for *Steds*. *Steds* System retrieved 142 tweets during a five hours time span (from 5:00 PM to 10:00 PM). However, most of which are retweets or tweets with similar contents. Then the tweets summarization algorithm was utilized. The results for *Steds* are listed in TABLE II.

TABLE II
TWEET SUMMARIZATION RESULTS AT APR. 26^{th} 2015 10:00 PM

Light Rail trains can't run right now because of **FreddyGray protesters** are on the tracks on Howard St @wbalradio
Leaving the @**Orioles game** - head SOUTH. Use 395 to 95 or Greene to 295. **DO NOT GO EAST OR NORTH** @wbalradio
Howard Street remains CLOSED north of I 395 @wbalradio
Heavier traffic on I 395 SOUTH leaving the city. Looks like @**Orioles fans** may be leaving early @wbalradio
From @mtamaryland - **ALL SERVICE** on Light Rail and Metro Subway has **resumed. Now normal. No delays** reported @wbalradio
Traffic stuck on Pratt St has been **cleared out**. Lombard is the biggest mess now. **Lombard CLOSED near Eutaw**. @wbalradio

Six summarized transportation related events have been extracted for April 26, 2015 10:00 PM: 1) Rail trains delay due to *Freddy Gray protest*. The tweet was originally posted by the user @*JimWBALTraffic* at 5:20 PM. And the report is proved to be true according to the new reports. 2) Heavy traffic on 295 SOUTH, due to *Baltimore Orioles* game. This tweet was originally posted by the user @*JimWBALTraffic* at 2: 10 PM. The event is verified according to the *Baltimore Orioles* schedule. 3) Street closure alert update. This tweet was originally posted by @*JimWBALTraffic* at 3: 30 PM. It was a following up street closure update. And the rest of the summarized tweets: 4) Heavy traffic alert update; 5) Rail train delay dismissed; and 6) heavy traffic report are updates for the previous extracted events. According to the results, *Steds* successfully summarized 142 tweets and generated a concise tweets summary of length 6. This shows the efficiency of the summarization performance.

IV. CONCLUSION

This paper presents *Steds*: an efficient and practical social media based traffic status monitoring system. Based on a set of influential users, a transportation related keywords extraction scheme is constructed. An outstanding query expansion algorithm for tweets is proposed. A tweet crawling process is then performed based on the expanded queries. A state-of-the-art tweets summarization algorithm is designed to eliminate the redundant tweets information. In addition, we show that the proposed tweets query expansion algorithm outperforms the previous methods. Case studies confirm the feasibility of summarizing the redundant tweets to generate concise textual contents.

ACKNOWLEDGMENT

This work is supported 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.

REFERENCES

[1] Y. Xie, Z. Chen, Y. Cheng, K. Zhang, A. Agrawal, W.-K. Liao, and A. Choudhary, "Detecting and tracking disease outbreaks by mining social media data," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 2958–2960.

[2] R. S. Jafarzadeh, "Emergency management 2.0 Integrating social media in emergency communications," *Journal of Emergency Management*, vol. 9, no. 4, pp. 15–18, 2011.

[3] A. Schulz, J. Ortmann, and F. Probst, "Getting user-generated content structured: Overcoming information overload in emergency management," in *Global Humanitarian Technology Conference (GHTC), 2012 IEEE*. IEEE, 2012, pp. 143–148.

[4] J. Cui, R. Fu, C. Dong, and Z. Zhang, "Extraction of traffic information from social media interactions: Methods and experiments," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 1549–1554.

[5] K. Fu, R. Nune, and J. X. Tao, "Social media data analysis for traffic incident detection and management," in *Transportation Research Board 94th Annual Meeting*. TRB, 2015.

[6] G. Erkan and D. R. Radev, "Lexrank: graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, pp. 457–479, 2004.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." 1999.

[8] 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*. ACM, 2014, pp. 557–560.

[9] K. Makice, *Twitter API: Up and running: Learn how to build applications with the Twitter API*. " O'Reilly Media, Inc.", 2009.

[10] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[11] I. Twitter. (2013) Rest apis — twitter developers. [Online]. Available: https://dev.twitter.com/rest/public