TSum4act: A Framework for Retrieving and Summarizing Actionable Tweets during a Disaster for Reaction

Minh-Tien NGUYEN[1], Asanobu KITAMOTO[2], and Tri-Thanh NGUYEN[3]

[1] Hung Yen University of Technology and Education (UTEHY)
tiennm@utehy.edu.vn
[2] National Institute of Informatics, Tokyo, Japan
kitamoto@nii.ac.jp
[3] Vietnam National University, Hanoi (VNU),
University of Engineering and Technology (UET)
ntthanh@vnu.edu.vn

Abstract. Social networks (e.g. Twitter) have been proved to be an almost real-time mean of information spread, thus they can be exploited as a valuable channel of information for emergencies (e.g. disasters) during which people need updated information for suitable reactions. In this paper, we present *TSum4act*, a framework designed to tackle the challenges of tweets (e.g. diversity, large volume, and noise) for disaster responses. The objective of the framework is to retrieve actionable tweets (e.g. casualties, cautions, and donations) that were posted during disasters. For this purpose, the framework first identifies informative tweets to remove noise; then assigns informative tweets into topics to preserve the diversity; next summarizes the topics to be compact; and finally ranks the results for user's faster scan. In order to improve the performance, we proposed to incorporate event extraction for enriching the semantics of tweets. *TSum4act* has been successfully tested on Joplin tornado dataset of 230.535 tweets and the completeness of 0.58 outperformed 17% of the retweet baseline's.

Keywords: Data Mining, Event Extraction, Tweet Summarization, Tweet Recommendation.

1 Introduction

Twitter provides a new method for spreading information during natural or man-made disasters [13] in the form of tweets (a short text message with the maximum of 140 letters). These tweets mention a wide range of information in all aspects of life, from personal aspects to disaster facts. During a disaster, tweets usually tend to explode in a large volume and high speed. This can challenge the people who seek up-to-date information for making decisions. To seek the

information, people can use Twitter's search function. However, this function bases on boolean queries and responses highly redundant tweets in a reverse chronological order. This challenge inspires us to present a framework that incorporates event extraction to generate event graphs in retrieving informative tweets for people in disasters.

Recently, tweet summarization has received a lot of attention from papers [5,12,8]. The authors in these researches have solved the summarization by using term frequency in corporation with inverted document frequency (TF-IDF) or lexical approach. However, these approaches face the noise of tweets and may not exploit entities (e.g. times, locations, numbers, etc) which play an important role in summarization [15]. These limitations inspire us to apply event extraction in our framework. Our contributions are:

- We adapt event extraction for improving the performance of summarization. Our approach has two advantages: (1) exploiting the important role of entities and (2) reducing the impact of lexical representation noise.
- We successfully apply our framework in a real dataset which is collected during Joplin tornado. The completeness measure of 0.58 indicates that information from our framework can be combined with other sources (i.e. TV, online news, or emergency services) to provide important information to people during disasters.

We set our problem as the tweet recommendation by relying on extractive summarization (e.g. based on event extraction). Given a disaster-related query, our model: 1) retrieves tweets containing the query from the source; 2) due to the fact that the tweets are diverse and noisy (e.g. there are a lot of unrelated ones), the model filters out irrelevant ones to get a smaller set of informative tweets; 3) in order to provide finer-grained information to users, the model divides informative tweets into predefined classes (i.e. casualties, cautions, and donations) ; 4) since the number of tweets in each class is still big, for each class of tweets, the model separates them into topics (in the form of clusters); finally, 5) to make the results compact, the model ranks the tweets and gets the top ones as a summary of each cluster to recommend to users. For improving the semantics and reducing the lexical noise of the tweets, we propose to represent tweets in the form of events (viz. after applying event extraction) for constructing a graph as the input for ranking algorithm.

The rest of this paper is organized as follows: related works are showed in Section 2; we will give our idea to solve informative tweets summarization in Section 3; our contributions are in Section 4.1, 4.2, 4.3, and 4.3; results are showed in Section 5; the last section is the conclusion.

2 Related Work

Ritter et al. [12] proposed a method which automatically extracted open domain events. In this method, events were ranked and classified before being plotted in calender entries. The result increased 14% of F1 over baseline (without NER).

However, the ranking mechanism can badly affect in generating result because the author used frequency of entities, thus unimportant entities may generate redundant events.

Chakrabarti and Punera used Hidden Markov Model to identify sub-events of the parent event with 0.5 of precision and 0.52 of recall in football matches [5]. To summarize information, the author combined *tf-idf* and *Cosine similarity*. However, this method might not achieve high precision because the noise of tweets. In our work, we rely on entities rather than single words (terms).

Khan et al. [8] used lexical level underlying topical modeling and graphical model for summarizing tweets in a debating event. The result was about 81.6% of precision and 80% recall with their dataset. However, this method faces the noise from tweets, does not utilize entities, and requires a large of lexicons for generating the summarization.

3 TSum4act Framework

TSum4act solves tweet summarization and recommendation in disaster responses. The problem can be defined as follows:

Input: the keywords related to a natural disaster.

Output: A small set of the most informative tweets which can be used in situational awareness for supporting the making of suitable reaction. The tweets are divided into classes (e.g. casualty, caution, and donation), and each class, in turn, is divided into topics represented by top ranked tweets.

We call this set as *informative tweets* or *actionable tweets* because these tweets provide useful information which help people making suitable decisions in a disaster.

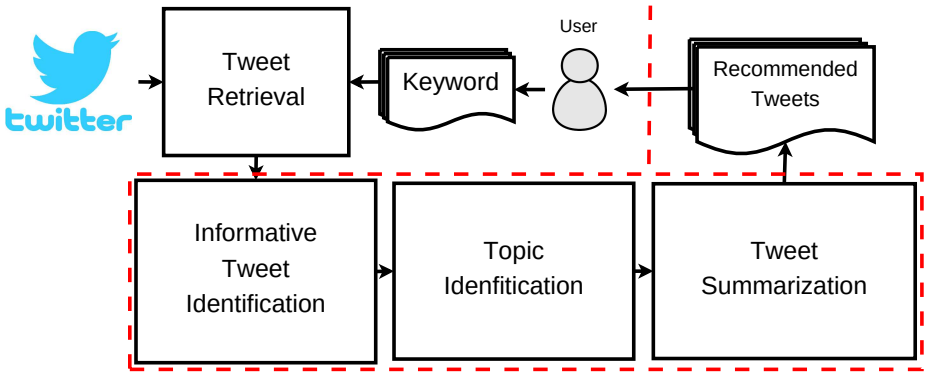


Fig. 1: Overview of TSum4act framework

The framework in Fig. 1 consists of four main components: tweet retrieval, informative tweet identification, topic identification, and tweet summarization.

The first component receives keywords from users and retrieves relevant tweets from Twitter. The second component identifies informative tweets (i.e. tweets which help people making decisions in a disaster rather than personal tweets). We solve this problem by using classification because tweets can be divided into *informative* and *not informative*. Another task of this component is to put informative tweets into important classes for users' easy navigation. The third component takes each tweet class to identify the different topics (i.e. tweets of which the meaning is close to each other) as a preliminary step to compress the tweets for reducing the tweet volume size. This component is built based on clustering which has the ability to group items that are close to each other into a cluster. The last component has the responsibility of summarizing the data to produce recommended tweets to users. This task is done by representing tweets in the form of events, and an event graph is constructed for each topic for ranking. After ranking, near-duplicate tweets are removed, and top ranked ones are returned to users as a summary for recommendation.

The first component is rather simple, therefore, in this paper we only focus on the three remaining components.

4 The Process of TSum4act

4.1 Informative Tweet Identification

We follow the approach of [7,14] to apply classification for informative tweet identification. The classification process includes three steps along with three binary classifiers. The first step distinguishes whether tweets are informative or not; the second one identifies whether informative tweets are direct (viz. events that users see or hear directly) or indirect, e.g. a tweet that is forwarded (called *retweet*); and the third step classifies the direct tweets into three classes: casualty/damage, caution/advice, and donation/offer (called valuable classes).

4.2 Topic Identification

The number of tweets is still big after classifying, we propose to assign tweets into topics (in the form of clusters) (to ensure the diversity) for later summarization (to keep the result compact). Since common clustering methods (e.g. K-means) normally base on the frequency of words (e.g. tf-idf), thus they are not suitable for keeping the semantics of tweets whereas hidden topic models are a good selection to solve this issue. Assigning tweets into clusters helps users to easily navigate the information. This component first generates document distributions by LDA and secondly, it uses a clustering algorithm to assign tweets into topic clusters.

LDA: LDA generates a probability distribution of documents (tweets) over topics by using estimated methods. Tweets can be assigned into clusters by a clustering algorithm based on this distribution. We decide to use LDA since it

was adapted for short texts (e.g., tweets or messages) [8]. Another interesting aspect is that it not only solves the clustering problem but also generates topical words which can be used for abstractive summarization. For simplicity, we have adopted Latent Dirichlet Allocation (LDA)[4] [1].

Clustering: LDA assigns tweets into hidden topics which are represented by words/phrases. However, the framework expects tweets which belong to clusters. Therefore, a clustering algorithm is used to solve this issue. The idea of assigning tweets into clusters is to find out the closest distance from tweet t to cluster c_i by an iterative algorithm which uses Jensen-Shannon divergence.

4.3 Tweet Summarization

This component first extracts events in tweets in each topic to built a graph. Subsequently, the event graphs are the input for a ranking algorithm to find out important (i.e. highest score) events . Finally, after removing near-duplicate tweets, top ones are returned to users.

Event Extraction: As introduced previously, we use a ranking algorithm to find out informative tweets. In fact, normal ranking algorithms only rely on keywords or even topical statistic can not completely utilize the semantics of tweets, thus, the final result may be not good. We propose to use ranking approach in which a tweet has a correlation with another based on the similarity. The similarity can be denoted by using words or complete tweets. However, using words or complete tweets faces the noise of tweet (i.e. stop words, hashtags, or emoticons); hence, this can badly affect in calculating the similarity. As the contribution, we propose to use event extraction to represent the similarity between two tweets. Using event extraction has two advantages: 1) reducing the noise of tweet and 2) keeping the semantics of tweets. Therefore, the framework can achieve high accuracy in generating informative tweets.

We define an event as a set of entities in a tweet. An event includes *subject*, *event phrase*, *location*, and *number* as follows.

$$event = \{subject, event\ phrase, location, number\} \quad (1)$$

where *subject* answers the question WHAT (e.g. a tornado or a road) which is a cause or result; *event phrase* represents the action/effect of the subject; *location* answers WHERE the event occurs (e.g. Oklahoma); and *number* focuses to answer the question HOW MANY (e.g. the number of victims).

To extract above attributes we use NER tool of [11] which annotates the tweets with predefined tags, then, we parse the result to extract values of the tags corresponding to the attributes. We accept an event which does not have complete attributes. An example of an event from an original tweet is showed as below:

[4] http://jgibblda.sourceforge.net/

Original tweet: *"Tornado kills 89 in Missouri yesterday"*
Event: {*Tornado, kills, Missouri, 89*}

In this example, *Tornado* is the subject, *kills* is the event phrase, *Missouri* is the location, and *89* is the number of victims. Events in this step are input for generating event graphs in the next section.

Event Graph Construction: The event graphs require vertices and weights as inputs. In this graph, two vertices (two events) are connected by an edge with a weight. To identify the weight we consider two equations: Simpson and Cosine. However, in this case, Cosine is better. The compared intuition of two equations is showed in Table 1. In this table, the value of *0.0* of Simpson indicates that

Table 1: The illustration of two equations

Tweet	**Simpson**	**Cosine**
Tornado kills 89 in Missouri Tornado kills 89 in Missouri yesterday	0.0	0.912

two tweets are completely similar despite the difference of the word *"yesterday"* between the two sentences. In contrast, the value of Cosine equation is *0.912* giving two tweets are nearly completely similar. Therefore, we follow Cosine to calculate the weight of two events. Given two events A and B, Cosine equation is showed in Eq. (2).

$$sim(A, B) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{2}$$

The *sim(A, B)* may be equal 0 in some cases indicating the two events are totally different. Therefore, to make the result compact, two events are deemed to have relation if they have the similarity measure above zero, and an event is removed if it has no relation with any other event.

After identifying vertices and weights the framework generates graphs for the raking step. In this paper, a graph is defined as $G =< V, E, W >$ where:

- V: a set of vertices denoted as $V = \{v_1, v_2, ..., v_n\}$ where v_i^{th} corresponds to an *event* i^{th} and n is the number of events in each cluster.
- E: a set of edges denoted as $E = \{e_1, e_2, ..., e_m\}$ where e_j^{th} connects two vertices v_k^{th} and v_h^{th} in V.
- W: a weight matrix holding the weight (in term of similarity) of edges.

In this graph, the weight matrix is calculated by Eq. (2). As this calculation, two vertices are connected when the weight is greater than 0. The graph G is the input for ranking algorithm in the next section.

Ranking: The goal of our method is to retrieve the most useful information. To solve this task a random method can be used after clustering. However, this method retrieves tweets which may not have enough evidence to conclude whether they are informative. An alternative approach is to use a ranking algorithm. It is suitable with our objective because: 1) our goal is to retrieve top k of tweets in each cluster and 2) important tweets converge at central clusters after clustering; hence the ranking algorithm can be easy to find out informative tweets.

The framework uses PageRank [2] for retrieving informative tweets. This is because, firstly, it is a successful algorithm for ranking webpages. Secondly, PageRank ranks vertices to find out important vertices. This is similar with our goal in finding important events. The ranking mechanism is showed in E.q (3).

$$PR(v_j) = (1-d) + d \sum_{v_i, vj \in E} \frac{W_{i,j} \times PR(v_i)}{\sum_{v_i, v_k \in E} W_{ik}} \tag{3}$$

Equation (3) shows the process to calculate the rank of a node in a recursive mechanism by using power iteration. In this equation, d is the damping factor which is used to reduce the bias of isolated nodes in a graph.

Filtering: The final module retrieves informative tweets corresponding to important events. Though it is possible to apply filtering before ranking, we put it after ranking to avoid the situation where the filtering produces a so sparse event graph that the performance of the ranking is badly affected.

The filtering uses Simpson equation to remove near-duplicate tweets because they provide the same information with others. Two near-duplicate tweets contain a little difference of information (e.g. in Tab 1). The Simpson is showed in Eq. (4).

$$simp(t_1, t_2) = 1 - \frac{|S(t_1) \cap S(t_2)|}{min(|S(t_1)|, |S(t_2)|)} \tag{4}$$

where *S(t)* denotes the set of words in tweet *t*.

5 Experiments and Results

5.1 Experimental Setup

Data: Dataset is 230.535 tweets, which were collected during Joplin tornado in the late afternoon of Sunday, May 22, 2011[5] at Missouri. The unique tweets were selected by Twitter Streaming API using the hashtag *#joplin*. The dataset is a part of AIDR project [6].

[5] http://en.wikipedia.org/wiki/2011_Joplin_tornado

Parameter Setting: Following [8] we choose $\alpha = \beta = 0.01$ with 1.000 iterations and $k \in [2, 50]$ for LDA. k will be identified in 5.3. In the Section 4.2 we use $\mu = 0.9$. Following [2], we choose $d = 0.85$ for PageRank. Finally, Eq. (4) only keeps tweets which satisfy $simp(t_1, t_2) > 0.25$. The value 0.25 is achieved by running the experiment over many times.

5.2 Retweet Baseline Model

We use retweet model as a baseline [4]. This is because, firstly, it represents the importance of a tweet based on retweet. Intuitively, if a message receives many retweets, it can be considered an informative tweet. Secondly, the model is simple because it has already provided by Twitter.

5.3 Preliminary Results

Classification: To build binary classifiers we use Maximum Entropy (ME)[6] with N-gram features because ME has been a successful method for solving classification problem [10]. Another interesting aspect is that our data is sparse after pre-processing while ME is a suitable approach to deal with sparse data. We do not use hashtags, emoticons, or retweets because these features are usually used in emotional analysis rather than in classification.

The performance of classification is measured by the average of 10-folds cross validation. Results are showed in Table 2.

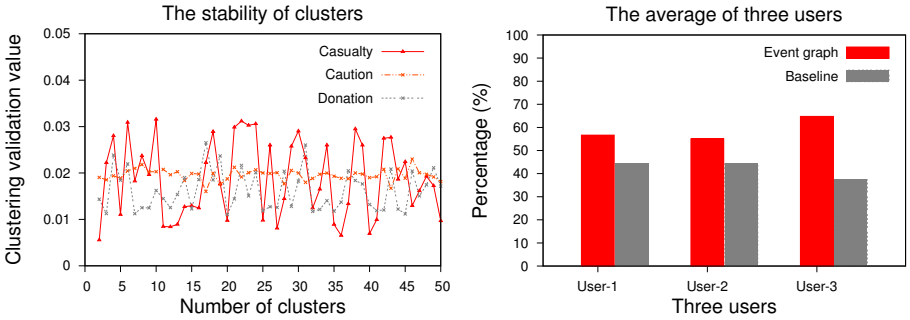
Table 2: The performance of classification

Class	**Precision**	**Recall**	**F1**
Informative Information	0.75	0.87	0.8
Direct Tweets	0.71	0.83	0.77
Casualty	0.89	0.89	0.89
Caution	0.88	0.91	0.89
Donation	0.87	0.88	0.88

The results in Tab. 2 show that classifiers achieve good performances 0.8, 0.77, and about 0.89 of F1 in the three classification levels, respectively. After classification, the framework only keeps tweets of the valuable classes including casualty/damage, caution/advice, and donation/offer.

Selecting the number of clusters: We follow cluster validation to select k [9,3]. We only select $k \in [2 : 50]$ because if $k > 50$ clusters may be overfitting while it is too general if $k < 2$ (only one cluster). The result of cluster validation is illustrated in Fig. 2a. After clustering, informative tweets belong to clusters which will be used for the summarization and recommendation.

[6] http://www.cs.princeton.edu/maxent



(a) The result of cluster validation (b) The average of completeness

Fig. 2: Selecting k and the average of completeness of three users

Evaluation Method: To evaluate the performance of the framework, we use three annotators to rate retrieved tweets (top 10 tweets). This is because: 1) selecting top k (k=10) is similar to recommendation and 2) our data needs user ratings to calculate performance of this component. The retrieved tweets are given for annotators to rate a value from 1 to 5 (5: good; 4: good; 3: acceptable; 2: poor; and 1: very poor). Each tweet is rated in three times; the score of a tweet is the average of rating scores after rounding. The average of inter-rater agreement is 87.6% after cross-checking the results of three annotators.

Tweet Summarization and Recommendation: We define the *completeness* to measure the performance of our method. The completeness measures how well the summary covers the informative content in the recommended tweets by the total rated scores over maximal score in each cluster as in E.q. (5).

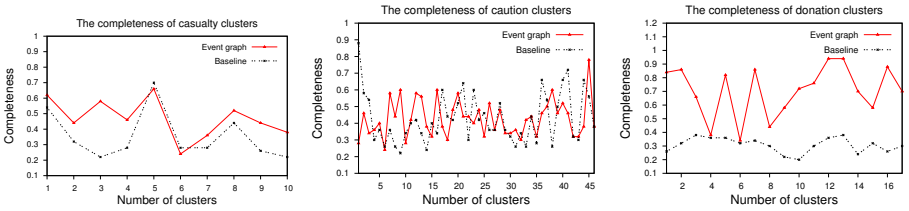
$$completeness = \sum (rating\ score)/50 \tag{5}$$

where *rating score* is the users' score, and 50 is the maximum total score (viz. 10 tweets each has a maximum score of 5).

The results of the three annotators are illustrated in Fig. 3, 4, and 5. indicating that our framework dominates the baseline in Fig. 3a, 3c, 4c, 5a, 5b, and 5c while it is hard to conclude our framework is better in Caution in Fig. 3b and 4b.

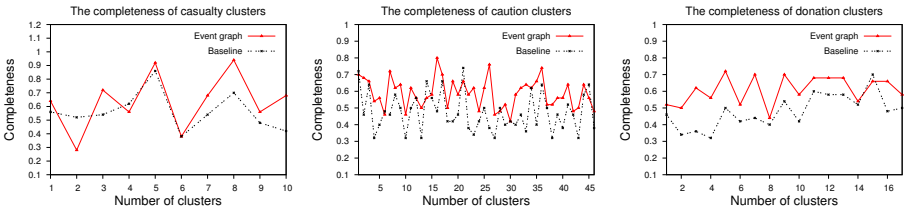
The average of completeness of three users is showed in Fig. 6. It is equal in Caution of 1^{st} annotator about 0.42 in Fig. 6a while our framework outperforms baseline on other annotators. In Fig. 6a, the result of our framework is twice higher than baseline in Donation about 0.7 and 0.3. The result of 2^{nd} annotator is higher than other users and the completeness of our method in Fig. 6b is around 0.6.

The average of completeness on three annotators over classes is showed in Fig. 2b. The result shows that the completeness of our method outperforms baseline



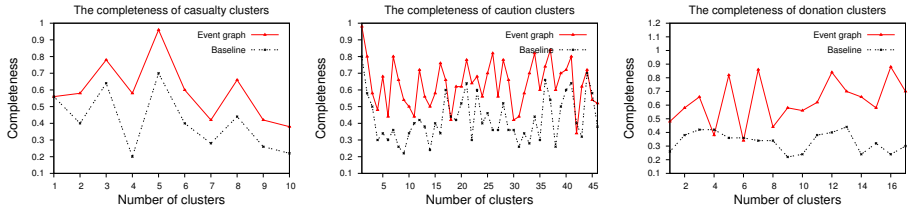
(a) The completeness of casualty (b) The completeness of caution (c) The completeness of donation

Fig. 3: The comparison of two methods in tweet recommendation of 1^{st} user



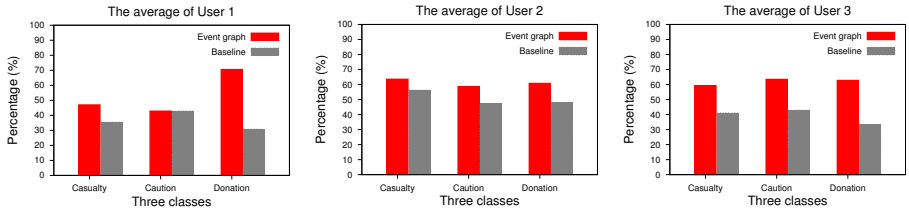
(a) The completeness of casualty (b) The completeness of caution (c) The completeness of donation

Fig. 4: The comparison of two methods in tweet recommendation of 2^{nd} user



(a) The completeness of casualty (b) The completeness of caution (c) The completeness of donation

Fig. 5: The comparison of two methods in tweet recommendation of 3^{rd} user



(a) The average of $1^{st}user$ (b) The average of $2^{nd}user$ (c) The average of $3^{rd}user$

Fig. 6: The average of completeness of two methods in tweet recommendation of three users

method about 17% (0.58 and 0.41, respectively). The results in Casualty and Caution are similar about 0.55 with our method and around 0.44 with baseline. In Donation our method outperformed the baseline with the result of 0.64 in comparison with 0.37.

5.4 Discussion

Fig. 3, 4, and 5 indicate that the results of our method are quite similar with baseline in Fig. 3b, 4a, 4b, and 5b, but in the remaining figures our method prevails.

By checking original tweets we recognize that clusters which have a high completeness contain highly relevant tweets. It appears in both the two methods. For example, almost top 10 tweets in cluster 1^{st} in 5b and 5^{th} in 5a are strongly related to the cautions, advices, or casualties of the tornado. By contrast, clusters which have a low completeness include many irrelevant tweets (including images and videos).

The checking process also shows that reasons making the performance of baseline is lower than that of our method. Firstly, there are many tweets which contain images or videos, but they are irrelevant according to our investigation. Secondly, many tweets in donation receive a lot of retweets because it mentions about a famous person (President Obama). Another interesting point is the speech of Eric Cantor, who is a member of GOP of U.S. Eric Cantor opposes disaster relief and saves money. Therefore, his tweets receive many retweets. Finally, a lot of tweets mention to memorial services for victims.

The results indicate that ranking based on event graph is an important factor which making our framework achieves a good performance. If an event receives a high weight, then it has many strong relations with others. It means that this event tends to appear in many tweets; hence, ranking events can retrieve important tweets. This phenomenon also appears in other clusters which have a high completeness. An alternative interesting point appears in cluster 5^{th} of caution in Fig. 3b. It is the lowest of completeness of our method. The checking process shows that many events are irrelevant indicating that the clustering and event extraction are not efficient in this cluster.

6 Conclusion

In this paper, we introduced *TSum4act*, a framework for retrieving informative tweets during a disaster for suitable reaction. Our framework utilizes state-of-the-art machine learning techniques, event extraction, and graphical model to deal with the diversity, large volume and noise of tweets. *TSum4act* was compared with retweet model and the result indicates that our framework can be used in real disasters.

For future works, we plan to first improve the performance of classification by trying multi-class classifiers. Secondly, the evaluation method should be considered because we only select the top 10 tweets. In addition, our method should

be compared with other model as lexicons. Finally, we plan to summarize recommended tweets by using abstractive approach.

References

1. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
2. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engines. *Computer networks and ISDN system*, 30.1:107–117, 1998.
3. Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics*, 2010.
4. Michael Busch, Krishna Gade, Brian Larson, Patrick Lok, Samuel Luckenbill, and Jimmy Lin. Earlybird: Real-time search at twitter. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on. IEEE*, 2012.
5. Deepayan Chakrabarti and Kunal Punera. Event summarization using tweets. In *ICWSM*, 2011.
6. M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg. Aidr: Artificial intelligence for disaster response. In *In Proceedings of the companion publication of the 23rd international conference on World wide web companion(pp. 159-162). International World Wide Web Conferences Steering Committee*, 2014, April.
7. Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Extracting information nuggets from disaster-related messages in social media. In *ISCRAM, Baden-Baden, Germany*, 2013.
8. Muhammad Asif Hossain Khan, Danushka Bollegala, Guangwen Liu, and Kaoru Sezaki. Multi-tweet summarization of real-time events. In *Proc. of ASE/IEEE International Conference on Social Computing. IEEE*, 2013.
9. Erel Levine and Eytan Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13.11:2573–2593, 2001.
10. Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering*, Vol. 1:61–67, 1999.
11. Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics*, 2011.
12. Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. Open domain event extraction from twitter. In *KDD, pp: 1104-1112*, 2012.
13. Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web. ACM*, 2010.
14. Sarah E. Vieweg. *Situational Awareness in Mass Emergency: A Behavioral and Linguistic Analysis of Microblogged Communications*. PhD thesis, University of Colorado at Boulder, 2012.
15. Wei Xu, Ralph Grishman, Adam Meyers, and Alan Ritter. A preliminary study of tweet summarization using information extraction. In *NAACL*, 2013.