

Civique: Using Social Media to Detect Urban Emergencies

Diptesh Kanojia^{*1,2,3}, Vishwajeet Kumar^{*1,2,3}, and Krithi Ramamritham¹

¹ IIT Bombay, India,

² Monash Univerity, Australia,

³ IITB-Monash Research Academy, India

`diptesh,vishwajeet,krithi@cse.iitb.ac.in`

Abstract. We present the Civique system for emergency detection in urban areas by monitoring micro blogs like Tweets. The system detects emergency related events, and classifies them into appropriate categories like “fire”, “accident”, “earthquake”, etc. We demonstrate our ideas by classifying Twitter posts in real time, visualizing the ongoing event on a map interface and alerting users with options to contact relevant authorities, both online and offline. We evaluate our classifiers for both the steps, i.e., emergency detection and categorization, and obtain F-scores exceeding 70% and 90%, respectively. We demonstrate Civique using a web interface and on an Android application, in realtime, and show its use for both tweet detection and visualization.

Keywords: Social Media, Event Detection, Data Analysis, Emergency Detection, Urban Emergency, Twitter

1 Introduction

With the surge in the use of social media, micro-blogging sites like Twitter⁴, Facebook⁵, and Foursquare⁶ have become household words. Growing ubiquity of mobile phones in highly populated developing nations has spurred an exponential rise in social media usage. The heavy volume of social media posts tagged with users’ location information on micro-blogging website Twitter presents a unique opportunity to scan these posts. These Short texts (e.g. ”tweets”) on social media contain information about various events happening around the globe, as people post about events and incidents alike. Conventional web outlets provide emergency phone numbers (i.e. 100, 911), etc., and are fast and accurate. Our system, on the other hand, connects its users through a relatively newer platform i.e. social media, and provides an alternative to these conventional methods. In

* These two authors contributed equally

⁴ <http://www.twitter.com>

⁵ <http://www.facebook.com>

⁶ <http://foursquare.com>

case of their failure or when such means are busy/occupied, an alternative could prove to be life saving.

These real life events are reported on Twitter with different perspectives, opinions, and sentiment. Every day, people discuss events thousands of times across social media sites. We would like to detect such events in case of an emergency. Some previous studies[10] investigate the use of features such as keywords in the tweet, number of words, and context to devise a classifier for event detection. [4] discusses various techniques researchers have used previously to detect events from Twitter. [9] describe a system to automatically detect events about known entities from Twitter. This work is highly specific to detection of events only related to known entities. [2] discuss a system that returns a ranked list of relevant events given a user query.

Several research efforts have focused on identifying events in real time([5][7][1][10]). These include systems to detect emergent topics from Twitter in real time ([5] [3]), an online clustering technique for identifying tweets in real time [7], a system to detect localized events and also track evolution of such events over a period of time [1]. Our focus is on detecting urban emergencies as events from Twitter messages. We classify events ranging from natural disasters to fire break outs, and accidents. Our system detects whether a tweet, which contains a keyword from a pre-decided list, is related to an actual emergency or not. It also classifies the event into its appropriate category, and visualizes the possible location of the emergency event on the map. We also support notifications to our users, containing the contacts of specifically concerned authorities, as per the category of their tweet.

The rest of the paper is as follows: Section 2 provides the motivation for our work, and the challenges in building such a system. Section 3 describes the step by step details of our work, and its results. We evaluate our system and present the results in Section 4. Section 5 showcases our demonstrations in detail, and Section 6 concludes the paper by briefly describing the overall contribution, implementation and demonstration.

2 Motivation and Challenges

In 2015, 53% of all unnatural deaths in India were caused by accidents, and 6% by accidental fires⁷. Moreover, the Indian subcontinent suffered seven earthquakes in 2015, with the recent Nepal earthquake alone killing more than 9000 people and injuring 23,000⁸. We believe we can harness the current social media activity on the web to minimize losses by quickly connecting affected people and the concerned authorities. Our work is motivated by the following factors, (a) Social media is very accessible in the current scenario. (The “Digital India” initiative by the Government of India promotes internet activity, and thus a pro-active social

⁷ <http://www.indiaspend.com/cover-story/fire-accidents-kill-54-every-day-yet-deaths-have-declined-81400>

⁸ https://en.wikipedia.org/wiki/April_2015_Nepal_earthquake

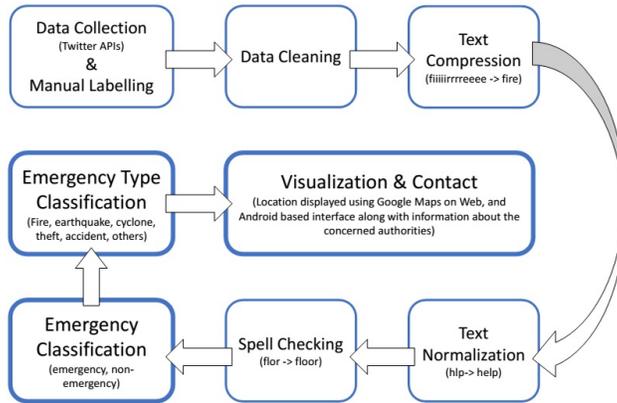


Fig. 1. System Architecture

Input	@user heelllllppp!!! Firreee at powai, lake lucene bldng 4th flor, hlp! #fire
Output	heelllllppp Firreee at powai , lake lucene bldng 4th flor , hlp

Table 1. Cleaning Script Output

media.) (b) As per the Internet trends reported in 2014⁹, about 117 million Indians are connected to the Internet through mobile devices. (c) A system such as ours can point out or visualize the affected areas precisely and help inform the authorities in a timely fashion. (d) Such a system can be used on a global scale to reduce the effect of natural calamities and prevent loss of life.

There are several challenges in building such an application: (a) Such a system expects a tweet to be location tagged. Otherwise, event detection techniques to extract the spatio-temporal data from the tweet can be vague, and lead to false alarms. (b) Such a system should also be able to verify the user’s credibility as pranksters may raise false alarms. (c) Tweets are usually written in a very informal language, which requires a sophisticated language processing component to sanitize the tweet input before event detection. (d) A channel with the concerned authorities should be established for them to take serious action, on alarms raised by such a system. (e) An urban emergency such as a natural disaster could affect communications severely, in case of an earthquake or a cyclone, communications channels like Internet connectivity may get disrupted easily. In such cases, our system may not be of help, as it requires the user to be connected to the internet. We address the above challenges and present our approach in the next section.

3 Our Approach

We propose a software architecture for Emergency detection and visualization as shown in figure 1. We collect data using Twitter API¹⁰, and perform language pre-processing before applying a classification model. Tweets are labelled manually with <emergency> and <non-emergency> labels, and later classified manually to provide labels according to the type of emergency they indicate. We use the manually labeled data for training our classifiers.

We use traditional classification techniques such as Support Vector Machines(SVM), and Naive Bayes(NB) for training, and perform 10-fold cross validation to obtain f-scores. Later, in real time, our system uses the Twitter streaming APIs to get data, pre-processes it using the same modules, and detects emergencies using the classifiers built above. The tweets related to emergencies are displayed on the web interface along with the location and information for the concerned authorities. The pre-processing of Twitter data obtained is needed as it usually contains ad-hoc abbreviations, phonetic substitutions, URLs, hashtags, and a lot of misspelled words. We use the following language processing modules for such corrections.

3.1 Pre-Processing Modules

We implement a cleaning module to automate the cleaning of tweets obtained from the Twitter API. We remove URLs, special symbols like @ along with the user mentions, Hashtags and any associated text. We also replace special symbols by blank spaces, and inculcate the module as shown in figure 1.

An example of such a sample tweet cleaning is shown in table 1.

While tweeting, users often express their emotions by stressing over a few characters in the word. For example, usage of words like *hellpppp*, *fiiiiireeee*, *ruuuuunnnnn*, *druuuuunnnkkk*, *soooooooo* actually corresponds to *help*, *fire*, *run*, *drunk*, *so* etc. We use the compression module implemented by [8] for converting terms like “pleeeeeeeeeaaaaasseeeee” to “please”.

It is unlikely for an English word to contain the same character consecutively for three or more times. We, hence, compress all the repeated windows of character length greater than two, to two characters. For example “pleeeeeaaaaassee” is converted to “pleeaassee”. Each window now contains two characters of the same alphabet in cases of repetition. Let n be the number of windows, obtained from the previous step. We, then, apply brute force search over 2^n possibilities to select a valid dictionary word.

Table 2 contains sanitized sample output from our compression module for further processing.

Text Normalization is the process of translating ad-hoc abbreviations, typographical errors, phonetic substitution and ungrammatical structures used in

⁹ “Internet trends 2014 report” by Mary Meeker, Kleiner Perkins Caufield & Byers (KPCB)

¹⁰ <https://dev.twitter.com/overview/documentation>

Sample Input	Sample Output
fiiiiireeeeee	fire
helllllp m stttuuuuuccckk	help m stuck

Table 2. Sample output of Compression module

Input Sentence	Output Sentence
bldng 4th flor, hlp	building 4th flor, help

Table 3. Sample output of Normalization module

text messaging (Tweets and SMS) to plain English. Use of such language (often referred as Chatting Language) induces noise which poses additional processing challenges.

We use the normalization module implemented by [8] for text normalization. Training process requires a Language Model of the target language and a parallel corpora containing aligned un-normalized and normalized word pairs. Our language model consists of 15000 English words taken from various sources on the web.

Parallel corpora was collected from the following sources:

1. Stanford Normalization Corpora which consists of 9122 pairs of un-normalized and normalized words / phrases.
2. The above corpora, however, lacked acronyms and short hand texts like *2mrw*, *l8r*, *b4*, *hlp*, *flor* which are frequently used in chatting. We collected 215 pairs un-normalized to normalized word/phrase mappings via crowd-sourcing.

Table 3 contains input and normalized output from our module.

Users often make spelling mistakes while tweeting. A spell checker makes sure that a valid English word is sent to the classification system. We take this problem into account by introducing a spell checker as a *pre-processing* module by using the JAVA API of Jazzy spell checker¹¹ for handling spelling mistakes.

An example of correction provided by the Spell Checker module is given below:-

Input: *building 4th flor, help*

Output: *building 4th floor, help*

Please note that, our current system performs compression, normalization and spell-checking if the language used is English. The classifier training and detection process are described below.

3.2 Emergency Classification

The first classifier model acts as a filter for the second stage of classification. We use both SVM and NB to compare the results and choose SVM later for stage

¹¹ <http://sourceforge.net/projects/jazzy/>

Classifier	Stage 1	Stage 2
SVM	72.5%	90.5%
NB	67.9%	92%

Table 4. Classification Results

one classification model, owing to a better F-score. The training is performed on tweets labeled with classes $\{emergency_i\}$, and $\{non-emergency_i\}$ based on *unigrams* as features. We create word vectors of strings in the tweet using a filter available in the WEKA API[6], and perform cross validation using standard classification techniques.

3.3 Type Classification

We employ a multi-class Naive Bayes classifier as the second stage classification mechanism, for categorizing tweets appropriately, depending on the type of emergencies they indicate. This multi-class classifier is trained on data manually labeled with classes. We tokenize the training data using “NgramTokenizer” and then, apply a filter to create word vectors of strings before training. We use “trigrams” as features to build a model which, later, classifies tweets into appropriate categories, in real time. We then perform cross validation using standard techniques to calculate the results, which are shown under the label “Stage 2”, in table 4.

3.4 Location Visualizer

We use Google Maps Geocoding API¹² to display the possible location of the tweet origin based on longitude and latitude. Our visualizer presents the user with a map and pinpoints the location with custom icons for earthquake, cyclone, fire accident etc. Since we currently collect tweets with a location filter for the city of “Mumbai”, we display its map location on the interface. The possible occurrences of such incidents are displayed on the map as soon as our system is able to detect it.

We also display the same on an Android device using the WebView functionality available to developers, thus solving the issue of portability. Our system displays visualization of the various emergencies detected on both web browsers and mobile devices.

4 Evaluation

We evaluate our system using automated, and manual evaluation techniques. We perform 10-fold cross validation to obtain the F-scores for our classification systems. We use the following technique for dataset creation. We test the system

¹² <https://developers.google.com/maps/documentation/geocoding>

in realtime environments, and tweet about fires at random locations in our city, using test accounts. Our system was able to detect such tweets and detect them with locations shown on the map.

4.1 Dataset Creation

We collect data by using the Twitter API for saved data, available for public use. For our experiments we collect 3200 tweets filtered by keywords like “fire”, “earthquake”, “theft”, “robbery”, “drunk driving”, “drunk driving accident” etc. Later, we manually label tweets with <emergency>and <non-emergency>labels for classification as stage one. Our dataset contains 1313 tweet with *positive* label <emergency>and 1887 tweets with a *negative* label <non-emergency>. We create another dataset with the positively labeled tweets and provide them with category labels like “fire”, “accident”, “earthquake” etc.

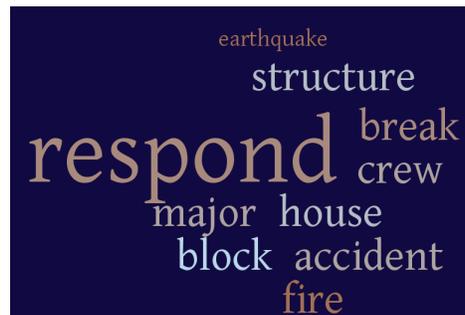


Fig. 2. Word Cloud of top attributes

4.2 Classifier Evaluation

The results of 10-fold cross-validation performed for stage one are shown in table 4, under the label “Stage 1”. In table 4, For “Stage 1” of classification, F-score obtained using SVM classifier is 72.5% as shown in row 2, column 2. We also provide the system with sample tweets in real time and assess its ability to detect the emergency, and classify it accordingly. The classification training for Stage 1 was performed using two traditional classification techniques SVM and NB. SVM outperformed NB by around 4% and became the choice of classification technique for stage one.

Some false positives obtained during manual evaluation are, “I am sooooo so drunk right nowwwwwww” and “fire in my office , the boss is angry”. These occurrences show the need of more labeled gold data for our classifiers, and some other features, like Part-of-Speech tags, Named Entity recognition, Bigrams, Trigrams etc. to perform better.

The results of 10-fold cross-validation performed for stage two classification model are also shown in table 4, under the label “Stage 2”. The training for

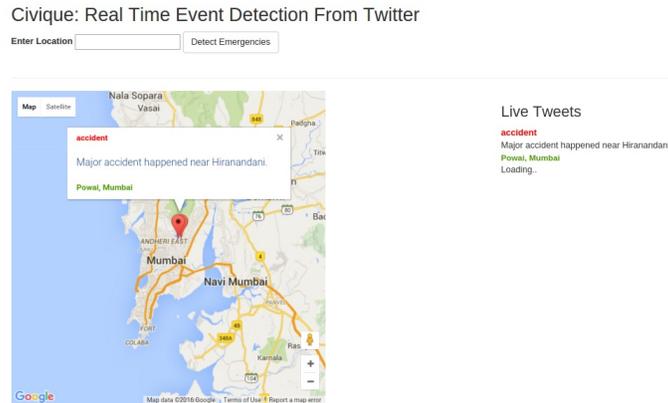


Fig. 3. Screenshot: Web Interface

stage two was also performed using both SVM and NB, but NB outperformed SVM by around 1% to become a choice for stage two classification model.

We also perform attribute evaluation for the classification model, and create a word cloud based on the output values, shown in figure 2. It shows that our classifier model is trained on appropriate words, which are very close to the emergency situations viz. “fire”, “earthquake”, “accident”, “break” (Unigram representation here, but possibly occurs in a bigram phrase with “fire”) etc. In figure 2, the word cloud represents the word “respond” as the most frequently occurring word as people need urgent help, and quick response from the assistance teams.

5 Demonstration Description

Users interact with Civique through its Web-based user interface and Android based application interface. The features underlying Civique are demonstrated through the following two show cases:

Show case 1: Tweet Detection and Classification

This showcase aims at detecting related tweets, and classifying them into appropriate categories. For this, we have created a list of filter words, which are used to filter tweets from the Twitter streaming API. These set of words help us filter the tweets related to any incident. We will tweet, and users are able to see how our system captures such tweets and classifies them. Users should be able to see the tweet emerge as an incident on the web-interface, as shown in figure 3 and the on the android application, as shown in figure 4. Figure 5 demonstrates how a notification is generated when our system detects an emergency tweet. When a user clicks the emerged spot, the system should be able to display the sanitized version / extracted spatio-temporal data from the tweet. We test the system in a realtime environment, and validate our experiments. We also report

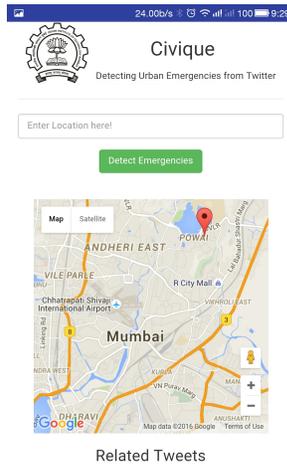


Fig. 4. Screenshot: Mobile Interface

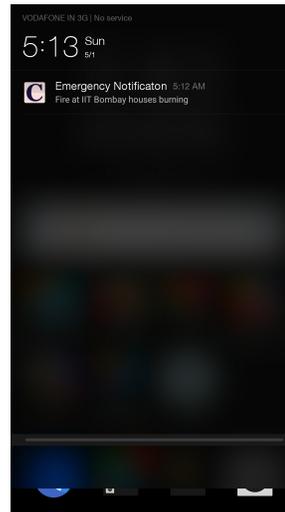


Fig. 5. Screenshot: Generated Notification

the false positives generated during the process in section 4.2 above.

Show case 2: User Notification and Contact Info.

Civique includes a set of local contacts for civic authorities who are to be / who can be contacted in case of various emergencies. Users can see how Civique detects an emergency and classifies it. They can also watch how the system generates a notification on the web interface and the Android interface, requesting them to contact the authorities for emergencies. Users can change their preferences on the mobile device anytime and can also opt not to receive notifications. Users should be able to contact the authorities online using the application, but in case the online contact is not responsive, or in case of a sudden loss of connectivity, we provide the user with the offline contact information of the concerned civic authorities along with the notifications.

6 Conclusions

Civique is a system which detects urban emergencies like earthquakes, cyclones, fire break out, accidents etc. and visualizes them on both on a browsable web interface and an Android application. We collect data from the popular micro-blogging site Twitter and use language processing modules to sanitize the input. We use this data as input to train a two step classification system, which indicates whether a tweet is related to an emergency or not, and if it is, then what category of emergency it belongs to. We display such positively classified tweets along with their type and location on a Google map, and notify our users to inform the concerned authorities, and possibly evacuate the area, if his location matches

the affected area. We believe such a system can help the disaster management machinery, and government bodies like Fire department, Police department, etc., to act swiftly, thus minimizing the loss of life.

Twitter users use slang, profanity, misspellings and neologisms. We use standard cleaning methods, and combine NLP with Machine Learning (ML) to further our cause of tweet classification. At the current stage, we also have an Android application ready for our system, which shows the improvised, mobile-viewable web interface.

In the future, we aim to develop detection of emergency categories on the fly, obscure emergencies like “airplane hijacking” should also be detected by our system. We plan to analyze the temporal sequence of the tweet set from a single location to determine whether multiple problems on the same location are the result of a single event, or relate to multiple events.

References

1. Abdelhaq, H., Sengstock, C., Gertz, M.: In: EvenTweet: online localized event detection from twitter (2013)
2. Agarwal, M.K., Bansal, D., Garg, M., Ramamritham, K.: Keyword search on microblog data streams: finding contextual messages in real time. In: Proceedings of 19th International Conference on Extending Database Technology (EDBT). pp. 15–18 (2016)
3. Alvanaki, F., Sebastian, M., Ramamritham, K., Weikum, G.: Enblogue: emergent topic detection in web 2.0 streams. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. pp. 1271–1274. ACM (2011)
4. atefeh, F.: A survey of techniques for event detection in twitter. In: computational intelligence
5. Cataldi, M., caro, L.D., schifanella, C.: Emerging topic detection on twitter based on temporal and social terms evaluation. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining (2010)
6. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter 11(1), 10–18 (2009)
7. Hila, B., Mor, N., Luis, G.: In: Beyond Trending Topics: Real-World Event Identification on Twitter (2011)
8. Kanojia, D., Dhuliawala, S., Gupta, N., Mishra, A., Bhattarcharya, P.: Transchat: Cross-lingual instant messaging for indian languages. In: Twelfth International Conference on Natural Language Processing. ICON (2015)
9. Popescu, A.M., Pennacchiotti, M., Paranjpe, D.: Extracting events and event descriptions from twitter. In: Proceedings of the 20th international conference companion on World wide web (2011)
10. Sakaki, T., Okazaki, M., Matsuo, Y.: In: Earthquake shakes Twitter users: real-time event detection by social sensors (2010)