EMERGENCY SITUATION NOTIFICATION BASED ON SOCIAL NETWORKS FOR MOBILE DEVICES

by

Hakan Anıt

B.S., Computer Engineering, Bahçeşehir University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2013

EMERGENCY SITUATION NOTIFICATION BASED ON SOCIAL NETWORKS FOR MOBILE DEVICES

APPROVED BY:

Suzan Üsküdarlı, Ph.D. (Thesis Supervisor)	.....
Assist. Prof. Arzucan Özgür	.....
Assist. Prof. Murat Şensoy	.....

DATE OF APPROVAL: 13.06.2013

ACKNOWLEDGEMENTS

I am deeply grateful to my thesis advisor, Ph.D. Suzan Üsküdarlı for all kinds of valuable assistance that she provided for this thesis to become reality. This thesis that I started as a course project, enriched by the creative ideas of my thesis advisor. She was always indulgent and directive throughout the times we have been working together.

I would like to thank my colleague Vildan Dursun for annotating thousands of sentences, Elif Zehra Türe Bakalcı for checking my thesis about typographical errors and Arda Çelebi for his clear thoughts about the project.

Besides all thanks, I am grateful to my family for their infinite love and support. You have always supported the decisions I take, always have been with me and lead me to the correct path. I owe you my success.

ABSTRACT

EMERGENCY SITUATION NOTIFICATION BASED ON SOCIAL NETWORKS FOR MOBILE DEVICES

There is rapid increase in the number of microbloggers everyday. Users can share their thoughts instantaneously. In this thesis, we have studied on identification of emergency related contributions in social media and mobile emergency notification. Our goal is to notify interested people immediately or beforehand about possible existence of an emergency situation. We are extracting emergency related information from microblogging systems with the help of supervised learning methods. Afterwards, we estimate the location of the incident from the messages we have collected from microblogging systems. Once we identify an incident occurrence, we send a notification to people nearby the incident via mobile application. We try to answer if contributors share emergency related information on microblogging systems and if so, how contributors express different types of incidents (ex: fire, earthquake) in microblogging systems. Whatever the incident type, location information is critical to determine. Our study exposes that, in case of emergency situations contributors share location information in post. In case contributor specifies an incorrect location in profile, we have found that in post location indicator points to the location incident occurred. We propose a model that fetches incident related posts from microblogging system, estimates incident location and notifies people nearby incident occurrence. We describe formal structure of our model and prototype developed. We evaluate our system by the accuracy of location, time estimation for incident detection about fire and earthquake incidents. Once we compare detections of our system with trustworthy sources, out of 246 earthquake incidents 25, out of 85 fire incidents 42 of them generated notifications. As a result of evaluation, we assess future improvements of our system.

ÖZET

MOBİL CİHAZLAR İÇİN SOSYAL MEDYA TABANLI ACİL DURUM BİLDİRİMİ

Mikroblog sistemlerinin kullanıcı sayıları gün geçtikce artmaktadır. Bu sistemlerde kullanıcılar düşüncelerini anlık olarak paylaşabilir. Tezde sosyal medya tabanlı acil durum tespit ve mobil bildirim üzerine çalışılmıştır. Amacımız herhangi bir olay yaşandıktan hemen sonra yada olay yaşanmadan önce o bölgede bulunan insanlara olayın varlığı konusunda bir bildirim yapabilmektir. Çalışmamızda gözetimli öğrenme metodları yardımıyla, mikroblog sisteminden acil durumu tasfir eden bilgi çıkartıyoruz. Topladığımız mesajlardan olayın konumunu tahmin ediyoruz. Tespit ettiğimiz olayın çevresinde bulunan kişilere, geliştirdiğimiz mobil uygulama ile bildirim gönderiyoruz. İnsanlar acil durumlarda yaşadıkları olayları mikroblog sistemlerde paylaşıyorlarmı, farklı türdeki olaylar (örn: yangın, deprem) mikroblog sistemlerde kullanıcılar tarafından nasıl paylaşılır bu sorulara cevap bulmaya çalışıyoruz. Olay her ne olursa olsun konum bilgisinin tespit edilmesi kritik öneme sahipir. Çalışmada görüyoruz ki acil durumlarda insanlar paylaştıkları mesajın içinde olayın yaşandığı konumuda paylaşmaktadırlar. Kişinin profilinde belirttiği konum bilgisi farklı bir bölgeyi işaret etsede, eğer kişi paylaşım yaptığı mesajın içinde konum belirtmişse aslında olayın tam olarak yaşandığı yada olaya yakın bir bölgeyi paylaştığını görüyoruz. Mikroblog sistemlerden olay ile alakalı mesajları toplayan, olayın nerede yaşandıgını tahmin eden ve olayın çevresinde bulunan kişilere bildirim yapacak bir model öneriyoruz. Önerdiğimiz modelin formal yapısını ve prototip uygulaması ortaya koyuyoruz. Çalışmamızı mikroblog sistemlerden deprem ve yangın olaylarını; zaman, konum bazında tespit etme başarısı ile değerlendiriyoruz. Sistemimizin çalıştığı zamanları güvenilir kaynaklarla karşılaştırdığımızda 246 depremden 25 tanesini, 85 yangından ise 42 tanesi hakkında bildirim yaptığını görüyoruz. Bu testin sonucuyla modelimizin gelişim alanlarını değerlendiriyoruz.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . x
LIST OF TABLES . . . xiii
1. INTRODUCTION . . . 1
1.1. Motivation . . . 1
1.2. Problem Statement . . . 3
1.3. Objective of this thesis . . . 5
1.4. Research questions . . . 6
1.5. Structure of this thesis . . . 6
1.6. Contribution . . . 7
2. BACKGROUND . . . 8
2.1. Twitter . . . 8
2.1.1. Followers . . . 8
2.1.2. Privacy . . . 8
2.1.3. Twitter slang . . . 9
2.1.4. Spam . . . 10
2.1.5. Usage statistics . . . 11
2.1.6. Twitter API . . . 13
2.1.6.1. REST API . . . 13
2.1.6.2. Search API . . . 13
2.1.6.3. Streaming API . . . 13
2.1.6.4. API parameters and format . . . 14
2.1.6.5. HTTP Error status codes for Twitter API . . . 14
2.2. Text classification . . . 15
2.2.1. Preprocessing . . . 15
2.2.1.1. Stop word removal . . . 15
2.2.1.2. Lemmatization . . . 16

2.2.1.3. POS Tagging 16
2.2.1.4. NER 17
2.2.2. Vector Space Model 17
2.2.3. Feature selection 18
2.2.4. N-grams 18
2.2.4.1. Collocation 18
2.2.5. Classification methods 19
2.2.5.1. Naive Bayes classifiers 19
2.2.5.2. Support Vector Machines 20
2.3. Linked data 20
2.3.1. About Linked data 20
2.3.2. DBPedia 21
2.3.3. Using Linked data in case of emergency 22
2.4. Non-Relational databases 23
2.4.1. NoSQL models 25
3. MODEL 27
3.1. Incident data sources 28
3.2. Incident characteristics and microblogger sharings 30
3.3. Classification 44
3.4. Incident detection 49
3.5. Location extraction 50
3.6. Notification 57
4. PROTOTYPE IMPLEMENTATION 60
4.1. Front-end 60
4.1.1. Web 61
4.1.1.1. Reporting 61
4.1.1.2. Search 62
4.1.1.3. Annotation 62
4.1.1.4. Training/Test sets 63
4.1.1.5. View Incidents 63
4.1.2. Mobile 63

4.1.3. Main technologies for Front-end 65
4.2. Back-end 69
4.2.1. Database 69
4.2.1.1. Design 69
4.2.1.2. Schema 70
4.2.2. Application Layer 74
5. EVALUATION 77
5.1. NER Location performance 77
5.2. Classification performance 78
5.3. Incidents detected 81
5.4. Implementation related experiments 97
5.4.1. Processing time of a single tweet 97
5.4.2. Notification medium experiments 102
6. DISCUSSION AND FUTURE WORK 103
6.1. Discussion 103
6.2. Contribution characteristics 107
6.3. Sample incident detections by EmNotifier and related tweets 112
6.4. Future work 115
6.4.1. Query expansion 115
6.4.2. Disambiguation 115
6.4.3. Performance 116
6.4.4. Incident detection 116
6.4.5. Vandalism 116
6.4.6. Classification 117
6.4.7. Handling global incidents 117
6.4.8. Reusable data 117
7. RELATED WORK 118
7.1. Great disasters and Social Media 118
7.2. Information about usage of Social Media in an Emergency Situation . . 119
7.3. Academic Studies 120
8. CONCLUSION 129

APPENDIX A: STOP WORDS LIST 130
APPENDIX B: MICROBLOG POST CLASSIFICATION PERFORMANCE . 133
APPENDIX C: WORD CLOUD FOR EARTHQUAKE, FIRE, RIOT 146
APPENDIX D: MICROBLOG POST TEMPLATES 148
APPENDIX E: MICROBLOG POST SAMPLES 154
REFERENCES 161

LIST OF FIGURES

Figure 2.1. Posting screen on Twitter. 9

Figure 2.2. Attach a location to tweet. 10

Figure 3.1. EmNotifier high level model. 27

Figure 3.2. An example of raw microblog post. 45

Figure 3.3. StreamReader algorithm to fetch query based posts from microblog system. 46

Figure 3.4. PrepareFeatureVector algorithm to prepare feature vector for classification. 47

Figure 3.5. An example of raw and processed microblog post text. 48

Figure 3.6. IncidentDetection algorithm to detect incidents. 50

Figure 3.7. IncidentCandidateCheck algorithm to determine if an incident candidate belongs to new or ongoing incident. 51

Figure 3.8. ExtractLocationFromFep algorithm to extract location from feature enriched post. 53

Figure 3.9. An example for possible Location_l,"Paris". 54

Figure 3.10. MatchLocations algorithm to match all possible combinations of Location. 55

Figure 3.11. ExtractLocationFromProfile algorithm to extract profile location from microblog user profile. 56

Figure 3.12. ExtractLocationFromPost algorithm to extract location from post. 58

Figure 3.13. Notifier algorithm to send notification to mobile users. 59

Figure 4.1. Prepare Customized Report in front-end web interface. 61

Figure 4.2. Show Latest Report in front-end web interface. 61

Figure 4.3. Search Tweets in front-end web interface. 62

Figure 4.4. Annotation process in front-end web interface. 62

Figure 4.5. Show interval annotated in front-end web interface. 62

Figure 4.6. Annotate interval in front-end web interface. 63

Figure 4.7. Show tweets annotated in front-end web interface. 63

Figure 4.8. Training/Test set preparation in front-end web interface. 64

Figure 4.9. View incidents in front-end web interface. 64

Figure 4.10. Mobile Notification. 64

Figure 4.11. Mobile Front-end in case of earthquake incident. 66

Figure 4.12. Mobile Front-end in case of fire incident. 67

Figure 4.13. Mobile Application Customization. 68

Figure 4.14. EmNotifier database design. 70

Figure 5.1. Location detected by EmNotifier for fire incident. 85

Figure 5.2. Location detected by EmNotifier for earthquake incident. 86

Figure 5.3. Importance of notification fire. 87

Figure 5.4. Notified area sample-1 in case of earthquake. 88

Figure 5.5. Notified area sample-2 in case of earthquake. 89

Figure 5.6. Notified area sample-1 in case of fire. 90

Figure 5.7. Notified area sample-2 in case of fire. 91

Figure 6.1. Contributors reporting earthquake. 108

Figure 6.2. Contributors reporting fire. 109

Figure 7.1. Count of “fire” query word related posts from microblogging system in 5 minutes interval. 122

Figure C.1. Frequently used words for earthquake incident. 146

Figure C.2. Frequently used words for fire incident. 146

Figure C.3. Frequently used words for riot incident. 147

LIST OF TABLES

Table 3.1. PrimitiveDT: Primitive data types for handling microblogs. 32

Table 3.2. MicroblogDT: Data types of a microblogging system. 34

Table 3.2. MicroblogDT: Data types of a microblogging system (cont.). 35

Table 3.3. MicroblogProcessingDT: Data types for processing microblogs. . . 35

Table 3.4. EmNotifier ProcessingDT: Data types for EmNotifier. 36

Table 3.5. Model Constants. 36

Table 3.6. Text processing related functions. 37

Table 3.7. FeatureEnrichedPost related functions. 38

Table 3.8. Interacting with Microblogging system related functions. 39

Table 3.9. Classification related functions. 39

Table 3.10. Incident detection related functions. 40

Table 3.10. Incident detection related functions (cont.). 41

Table 3.11. Location extraction related functions. 42

Table 3.12. Notification related functions. 43

Table 3.13. Types of location related elements in FeatureEnrichedPost. 52

Table 3.14. An Example for misspelling in microblogging systems. 57

Table 4.1. Java classes developed in EmNotifier. 76

Table 5.1. Sample NER annotations. 78

Table 5.2. Average F-score comparison of earthquake incident for Naive Bayes and SVM classifer. 79

Table 5.3. Average F-score comparison of fire incident for Naive Bayes and SVM classifer. 79

Table 5.4. Average F-score comparison of earthquake incident for Naive Bayes and SVM classifer with feature set-1 and feature set-2. 80

Table 5.5. Average F-score comparison of fire incident for Naive Bayes and SVM classifer with feature set-1 and feature set-2. 80

Table 5.6. Earthquake incident occurrences detected by EmNotifier and verified by trustworthy sources (24/5/13 10:20 - 28/5/13 21:54). 92

Table 5.7. Earthquake incident occurrences detected by EmNotifier and verified by trustworthy sources (30/5/13 21:37 - 1/6/13 14:49). 93

Table 5.8. Earthquake incident occurrences detected by EmNotifier and verified by trustworthy sources (1/6/13 16:28 - 1/6/13 19:35). 94

Table 5.9. Fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (24/5/13 22:50 - 25/5/13 00:00). 95

Table 5.10. Fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (25/5/13 12:00 - 27/5/13 15:43). 96

Table 5.11. Fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (27/5/13 17:46 - 27/5/13 20:00). 97

Table 5.12. Batch size=1. Avg. delay with location count 1=7.293 sec, Avg. delay with location count 2=9.655 sec, Avg. delay with location count 3=32.129 sec, Avg. delay with location count 4=39.191 sec. 99

Table 5.13. Batch size=5. Avg. delay with location count 1=15.926 sec, Avg. delay with location count 2=12.825 sec, Avg. delay with location count 3=13.946 sec, Avg. delay with location count 4=9.121 sec. . 100

Table 5.14. Batch size=10. Avg. delay with location count 1=27.694 sec, Avg. with location count 2=15.113 sec, Avg. delay with location count 3=16.705 sec, Avg. delay with location count 4=12.053 sec. 101

Table 5.15. Push notification delay on Public Wifi, 3G and Edge Average delay Public Wifi=5.407sec, Average delay Edge=7.0545, Average delay 3G=5.477. 102

Table 6.1. Tweet characteristics. 111

Table B.1. 2-fold cross validation, Train set count= 619, Test set count= 619, Positive tweets=396, Negative tweets=223,Min F-Score=0.839, Max F-Score=0.914, Avg. F-Score=0.877. 133

Table B.2. 2-fold cross validation using Naive Bayes classifer. Train set count= 619, Test set count= 619, Positive tweets=396, Negative tweets=223, Min F-Score=0.850, Max F-Score=0.865, Avg. F-Score=0.856. . . 133

Table B.3. 5-fold cross validation using SVM classifer. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93,Min F-Score=0.874, Max F-Score=0.907, Avg. F-Score=0.892. 134

Table B.4. 5-fold cross validation using Naive Bayes classifer. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93, Min F-Score=0.837, Max F-Score=0.889, Avg. F-Score=0.860. . . 134

Table B.5. 10-fold cross validation using SVM classifer. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Min F-Score=0.837, Max F-Score=0.939, Avg. F-Score=0.899. 135

Table B.6. 10-fold cross validation using Naive Bayes classifer. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Min F-Score=0.797, Max F-Score=0.913, Avg. F-Score=0.863. . . 136

Table B.7. 2-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 619, Test set count= 619, Positive tweets=386, Negative tweets=223, Avg. F-Score set-1=0.906, Avg. F-Score set-2=0.877. 136

Table B.8. 2-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 619, Test set count= 619, Positive tweets=386, Negative tweets=223, Avg. F-Score set-1=0.858, Avg. F-Score set-2=0.837. 137

Table B.9. 5-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93, Avg. F-Score set-1=0.892, Avg. F-Score set-2=0.824. 137

Table B.10. 5-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93, Avg. F-Score set-1=0.860, Avg. F-Score set-2=0.825. 138

Table B.11. 10-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Avg. F-Score set-1=0.910, Avg. F-Score set-2=0.899. 138

Table B.12. 10-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Avg. F-Score set-1=0.863, Avg. F-Score set-2=0.808. 139

Table B.13. 2-fold cross validation using SVM classifer with feature set-1 and-feature set-2. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Avg. F-Score set1=0.882, Avg. F-Score set2=0.834. 139

Table B.14. 2-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Avg. F-Score set-1=0.866, Avg. F-Score set-2=0.852. 140

Table B.15. 5-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Avg. F-Score set-1=0.895, Avg. F-Score set-2=0.845. 140

Table B.16. 5-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Avg. F-Score set-1=0.842, Avg. F-Score set-2=0.831. 141

Table B.17. 10-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Avg. F-Score set-1=0.861, Avg. F-Score set-2=0.821. 141

Table B.18. 10-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Avg. F-Score set-1=0.816, Avg. F-Score set-2=0.802. 142

Table B.19. 2-fold cross validation using SVM classifer. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Min F-Score=0.880, Max F-Score=0.884, Avg. F-Score=0.882. . . 143

Table B.20. 2-fold cross validation using Naive Bayes classifer. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Min F-Score=0.851, Max F-Score=0.880, Avg. F-Score=0.866. . . 143

Table B.21. 5-fold cross validation using SVM classifer. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Min F-Score=0.866, Max F-Score=0.912, Avg. F-Score=0.865. 143

Table B.22. 5-fold cross validation using Naive Bayes classifer. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Min F-Score=0.815, Max F-Score=0.852, Avg. F-Score=0.831. . . 144

Table B.23. 10-fold cross validation using SVM classifer. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Min F-Score=0.828, Max F-Score=0.897, Avg. F-Score=0.862. 144

Table B.24. 10-fold cross validation using Naive Bayes classifer. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Min F-Score=0.770, Max F-Score=0.846, Avg. F-Score=0.802. . . 145

Table D.1. Earthquake incident related user positive contribution templates. . 148

Table D.2. Earthquake incident related official content contribution templates. 149

Table D.3. Earthquake incident related negative contribution templates. . . . 150

Table D.4. Fire incident related user positive contribution templates. 151

Table D.5. Fire incident related official content contribution templates. 152

Table D.6. Fire incident related negative contribution templates. 153

Table E.1. Earthquake incident positive tweets. 154

Table E.2. Earthquake incident official content tweets. 155

Table E.3. Earthquake incident neutral tweets. 155

Table E.4. Fire incident positive tweets. 156

Table E.5. Fire incident official content tweets. 157

Table E.6. Fire incident neutral tweets. 158

Table E.7. Riot incident positive tweets. 159

Table E.8. Flood, Hurricane incident positive tweets. 160

1. INTRODUCTION

1.1. Motivation

Today the number of social networks users are progressively increasing. According to December 2012 figures, social networks like Facebook, Google+ and Twitter have users of 1.11Billion, 343M, and 500M, respectively [1]. Each social network has its own characteristics. A user of a Facebook and Google+ can create personal albums, add friends, send/receive messages and share data with different contents. Whereas Twitter is a microblogging system where a user can momentarily share his/her thoughts with a limit of 140 characters. Even though this system has a limit for characters, number of messages has not been limited. Different users use Twitter differently. Each user has his/her own characteristic writing style.

Twitter has more than 200M active users and these users daily share 177M [2] messages on average. Very valuable information can be gathered by analyzing messages of such a microblogging system which is so intensively used by people.

Our researches show that researchers intensively use the messages in Twitter system for different types of studies and promising results are being obtained. Bollen *et al.* [3], found a correlation between the moods of Twitter users and coming changes in stock markets. According to this study, the developed system can predict whether stock value of an organization will be gain or loss in a few days' period with a success of 86.7%. Bermingam *et al.* [4] has modeled the vote tendency of the people in the 2011 Ireland government selections by using the social media messages. According to the study, which party people would vote for, in the elections period could be predicted by examining the social media messages. Similar studies are done for 2008 U.S presidental debate (Diakopoulos *et al.* [5]), and 2011 French presidental election (Bouillot *et al.* [6]). Such kind of studies show that, Twitter can be used to predict the results of political elections.

Twitter messages are also used by the researchers for determination of emergencies and collection of information after disasters. Neubig *et al.* [7] has worked on collecting safety information after disaster for 2011 East Japan earthquake by using Twitter messages. By the help of the developed system, they identified the safety information of 100 people (he/she is living, seen at a certain location). Their system design can be used to provide safety related information for greater disasters.

In this thesis, we worked on social media based emergency determination and mobile notification. We use Twitter as the social media system. Howsoever we don't want big disasters (earthquakes, tsunami, landslides), fires, terrorist scenes, traffic accidents to occur, such kind of natural or unnatural (related to human) disasters can't be avoided. Unfortunately, although researchers work hard to be able to determine the natural disasters before they occur, we don't have such systems definitely to alarm people beforehand. Besides natural disasters, traffic accidents, domestic fires by human negligence or human life threats from people with bad intentions can be determined in advance according to the type of the event (for example: fire occurring as a result of inaccurate cable laying can be avoided if necessary controls are made by authorities) or it is a problem of intelligence (example: terrorist attacks). Whatever the situation is, it is not possible to predict both of the events beforehand.

Every incident can give results threatening human life at the end. More lives can be saved from a house fire or a bombing event by preventing people entering these areas. At this stage, aim of our study is to be able to make a notification about the event for the people in the region where the event takes place just after it occurs. This notification can be perceived in two different ways. An ordinary person decides not to go to that area. If the person is capable of answering that emergency (police, doctor etc), he/she can go to the accident scene and can help the people over there depending on the type of the event. He/she can provide the victims the need they require.

We can save more lives if we can send a message which more people will understand. Every person has his/her own characteristics, some might be illiterate, some may be blind, disabled or deaf. Taking into account such peculiarities of people, notice

sent should be understood by everyone.

In this respect, we extract emergency related information by supervised learning methods from Twitter. We estimate the location of the event by the help of posts and make it definite with DBPedia and Open Street Maps. We send notification to the people around the identified event by the mobile application we have developed. We are trying to find answers to such questions as "Do people share in Twitter what they live in emergency situations?", "Do different characteristics events (like fire, earthquake) have different ways of expression in Twitter?", "How can the location of an event be most accurately estimated by the help of social media posts?". We realize that people attach a picture of the scene (URL of the website in which picture is loaded) at the end of post if it is a fire post, if it is an earthquake the post is much shorter. Whatever the event is, it is critically important to determine the location information. In our study, we see that, in emergency situations, people also share the location along with the post they share. Although the information in the personal profile show a different location, we see that he/she shares the real location of the event if he/she specifies the location in his/her post.

1.2. Problem Statement

In this project, our primary target is to determine the users of the social media experiencing the event, to be able to determine the presence of event by the help of the information shared by those users and notify interested users in the event region. Considering the whole picture, there will be certainly a need for a help after an event occurs. But we can't give a proper response if we can't determine the event and its facts. So determination of the event is an important phase for response.

An ordinary person may not be authorized to interfere an event. For these kind of people the important information is the kind of event in the region (for example: fire, earthquake, tsunami), whether the event is still going on or not and the exact location. For example, a businessman might postpone his travel to the fire area in "A Street" until it has been completely finished. For this person, it is important to know

whether the incident is still active or not.

The main problem we are trying to solve in this thesis, is to determine the location and the type of the events occurring in different places on earth in real-time by the help of posts shared in social media. For this, it is preferably important to determine the posts of the users experiencing the event by himself/herself and sharing it in social media. After that, estimation of the location of the event as detailed as possible is necessary. If it is an earthquake, determination of the location as a country or a city may be efficient whereas if it is a fire, determination of the street is also required. Although we have concentrated on earthquakes and fires as sample events in this thesis, our model can be applied on different kinds of events.

Another problem and target of ours, is to send information of the event we have determined to our users around the region as fast as possible. Classifying the events, earthquake is a big- scaled event, whereas fire or a car accident happen in small scales. People can get detailed information about an earthquake from TV channels. But TV channels or other large scale media may not share every event. For example a car fire in “A street” may not be shown in news channels. Determination of such kind of small scale events is important to be known for the people living in the region or for the people intending to go there.

Another important problem is related with characteristics of people. System is sending a simple message about the location and type of the event to the users. This message should be customized so that it could be understood easily by different people. The situation should be explained vocally to a blind person or to a person who doesn’t know how to read. If the user is deaf it should be explained written or graphically.

While sending notifications to the users, it is a problem to determine the location of the users that the notification will be sent. If it is an earthquake, considering the propagation speed and severity of it, it could be necessary to inform a bigger population. In case of a fire, considering the direction of the wind, people in this direction should be informed.

Sometimes more than one event may occur at a certain location at the same time. It is a problem to determine all these events in real-time and rapidly considering the conditions we have mentioned. Like in every other systems, it is possible that there could be users trying to deceive, pretending nonexistent events take place and trying to make chaos. Solving this trust problem will provide more users use this system and adopt it.

1.3. Objective of this thesis

Our objective in this thesis is to determine the incidents by social media in real-time, to extract the location of the incident with the highest sensitivity, aggregate user contributions for location estimation and send this information to the users as fast as possible. In this study, we describe formal structure, data types and functions of our model along with prototype implementation.

In order to extract event related posts from microblogging system, we have used supervised learning methods. We set up a training and testing set for different events to build up the classifier. While building this set, we annotated thousands of tweets manually. Annotation is not only useful for the training and testing sets but it also gives ideas about how people share emergency situations.

While making a location estimation, we used DBPedia and Open Street Map. Location information is vital for such a system to work in order to make the location identification correctly. During annotation process, we see that, users are beginning to share the information of the location in their posts. To get this information out of their messages makes it possible to make a better location estimation. As can be seen from our model, much useful information can be obtained by using the information that the user shares in his/her post using Linked data. For example in a post like “Bombing at Akmerkez mall!”, “Akmerkez”’ can be detected as the organization by NER. After that, by the help of Linked data detailed location information can be obtained from it.

After identifying the event and determining the location a notification should be

sent to the users. Considering that people and events are always in motion, we send a mobile notification. By the IOS mobile application we have developed, users receive notifications from their mobile devices at a short period of time after the event. All these processes should be done in real-time and data should be kept in a rapid database system. For this we used a NoSQL database in our design.

Our aim is to build a good classifier with discriminating features and to send notification about the emergency to users as soon as possible. A prototype of this whole design and tools capable of analyzing Twitter data instantaneously have been developed.

1.4. Research questions

In this thesis, we try to find the answers to the following questions.

- How can unstructured data be used to determine the incidents?
- Are there any systems in where incidents are shared in real-time?
- Are there any characteristics to separate the incidents from each other?
- Can location of the incident be determined by social media system messages?
- Up to which sensitivity level (city, country, state, town) can the location information of the incident be determined from social media sharings?
- How can information regarding the occurring incident be sent to the users in the fastest way?

1.5. Structure of this thesis

In Chapter 2, important background about the concepts we have used throughout the thesis been described. Chapter 3 describes our proposed model for emergency notification based on social media. Formal representation of the model have been explained with data types and functions. In Chapter 4, details about the prototype implementation of our proposed model have been described. Evaluation of our model and results have been discussed in Chapter 5. In Chapter 6, we have discussed results

of evaluation and contribution characteristics in microblogging systems. Chapter 7 introduces several event related academic studies that uses microblogging systems as a source. Finally, Chapter 8 contains conclusions.

1.6. Contribution

We can present the below items as the results of this thesis.

(i) *Dataset.* Between the dates November 2012 - May 2013, on certain dates, 80M tweets have been collected including the keywords earthquake, fire, riot and hurricane. Not all tweets include the first moments of the events. For training and testing sets, approximately 3000 tweets have been manually annotated.

(ii) *Platform.* Our objective in this thesis, is to inform people as soon as possible about an emergency mentioned in stream media or not. This notification should only contain events in the region around where the person is or interested in. In this thesis, we demonstrated how to design such a platform and how to reach the users rapidly. We developed a prototype working in real-time. Besides, we built up a platform where we can analyze microblog posts.

(iii) *Scientific contribution.* A model has been proposed to classify the messages of different events by supervised machine learning methods. Our proposed model can be used for any event. We determined that by developing technology of mobile devices, location, which is the most important information of an event, are shared by the users in their posts. We looked to the fact that if location exists in tweet in our event detection model differently from alike models. We showed that, in emergency situations, people could be alarmed after a short period of time after the event by the help of social media. Even though the profile data of the users show different locations, we determined that the location information in the users posts they shared really pointed out the event. Depending on this valuable information, we developed a prototype of our model and demonstrated that our model works in real time. We analyzed that users share differently and selected features by paying attention to the fact that events have different characteristics.

2. BACKGROUND

In this section, fundamental knowledge about the subjects mentioned in our thesis is given. General information regarding social media system we've used, methods, data resources and structure of database are presented.

2.1. Twitter

In our thesis we used Twitter as social media system. Twitter is a micro blogging system in which users can share their thoughts in posts limited with 140 characters. Users can send their Twitter posts (a.k.a tweet) by third party applications in their smart phones, by SMS or web browsers. Twitter is a real-time working system. Breaking news, rumors, thoughts, anything you can imagine and interesting can be shared. You should be a member of the site to follow the sharings and follow the accounts you are interested in. If people you follow share messages, they appear in your page. You can think of this page as a newspaper always renewing itself. Page layout stays the same but the content enriches according to your choices with the received messages. Apart from the people you know, you can follow news channels, celebrities and companies you are interested in. Twitter system offers you new accounts it thinks you might be interested in.

2.1.1. Followers

Another important concept in Twitter jargon is"follower". You can follow any user you want in Twitter. When the user makes "status update" all his/her followers can see the post.

2.1.2. Privacy

Users don't have to make their accounts public. If their accounts are private, only the followers allowed by the user can see the status updates. So somebody not allowed

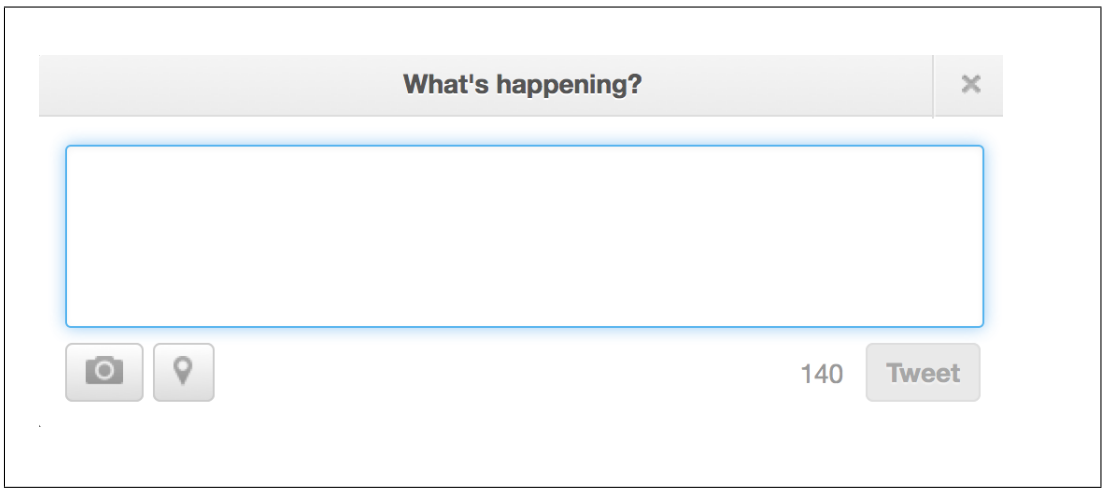


Figure 2.1. Posting screen on Twitter.

can't see the messages without permission. To follow a user with a private profile, you have to get his/her approval. Users can attach photos to their tweets. Every kind of EXIF data of the photos is excluded when it is uploaded into the system. So reaching private information like location by examining EXIF information in his/her photo is made impossible.

2.1.3. Twitter slang

Twitter has its own terms. We can show retweeting, mentioning and replying as examples. Retweeting (RT); makes it possible to be able to share the message you agree or like its content. Replying (reply); makes it possible to send a reply to a message. By Mentioning(username); you can refer to other users so you can start a new conversation with them. You can add a message or a video or share a link in your tweets. If you want your followers to know the place from where you are sharing a message, you can add your location to the message. You can automatically add your GPS position to your tweets if you set your application settings in your mobile instruments with GPS (see Figure 2.2). As there is a 140 character limit in Twitter, users develop different strategies in order to squash their thought in such a limited space. Users usually don't share their thoughts in a formal language. They always use abbreviations to complete their sentences. For example in Twitter you can see "smth hrrble hppned tdy" instead of "something horrible happened today". Users try not to use vowels as much as

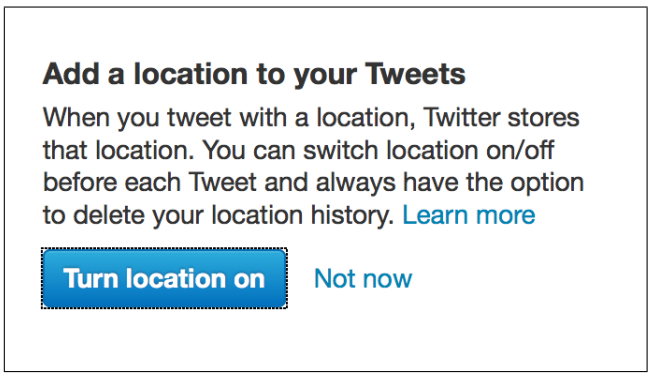


Figure 2.2. Attach a location to tweet.

possible to shorten their sentences. As it is not a newspaper where a formal language is needed, slang expressions are very common. Hashtags (#) are an indispensible part of Twitter system. Tagging a message you share is used to make it appear under a heading and make your message realized under this heading. For example: "fire in #Istanbul!". This message has been tagged with the word "Istanbul". By this way, if a user makes a search with the word "Istanbul" messages tagged with the word "Istanbul" will be listed. A sentence can have more than one hashtags. Moreover, nowadays, users began to tag every word in a sentence. Users can share news, photo or video from a different source by attaching its source link to the message. As links are usually long, URL shortener systems showed up. By this way users can attach the same link in a shortened way. For example the link with 77 characters `http://twittertoolsbook.com/10-awesome-twitter-analytics-visualization-tools/`, is shortened to 26 character long `http://tinyurl.com/3c97lty` by using TinyURL service.

2.1.4. Spam

Twitter system develops new tactics everyday to protect their users from spam messages and accounts. Spam messages are usually sent by a robot. Robot is registered in the system by a faked account. According to [8] users with spam intentions probably has the following characteristics;

- Posting harmful links (including links to phishing or malware sites).
- Aggressive following behavior (mass following and mass un-following for atten-

tion).

- Abusing the @reply or @mention function to post unwanted messages to users.
- Creating multiple accounts (either manually or using automated tools).
- Posting repeatedly to trending topics to try to grab attention.
- Repeatedly posting duplicate updates.
- Posting links with unrelated tweets.

Of course as it is the case with every system, Twitter may encounter some problems finding effective solutions against spam, but it constantly develops new solutions. By the help of artificial intelligence or other technics, spams can be sorted from regular users with a specific percentage of success. But human beings are more successful in that than the machines. Starting from this philosophy, Twitter has added a "Report as spam" button to each user's profile. With the help of this button, if users suspect that a user might be a spammer, they can report it to Twitter. By this operation, the user you mark as spammer can't reach you (can't follow or mention you). For more information, Twitter Help Center can be visited [8]. Of course tagging a user as a spammer doesn't guarantee that he/she will be kicked out from Twitter, but it is for sure that he/she can't communicate with you.

2.1.5. Usage statistics

Twitter publishes usage statistics in its own page. Many bloggers presumably publish data and statistics they have collected from various marketing companies. Twitter has started up on March 21st, 2006. According to April 2013 data [9], it has been reported that approximately 500M users are registered. 7 years have passed since its first operational day, and it is presumed that 170 Billion tweet have been sent up to date. On average, 400M tweets are sent daily.

We can't say that every single message in such a high volume sharing system hosts very important information. It can only be stated that by examining the unstructured data in such a high volume sharing system, useful information can be obtained. This useful data can be transformed into structured data. One of the hypotheses we propose

in our thesis is that everyday people are more and more accessing social media using mobile systems. Being directly proportional with this ratio, the possibility of finding detailed and accurate location information in the messages increases by the help of developing technology. According to our study, 40% of tweets are being sent by mobile devices. While these values vary according to the search keywords, we can say that ratio of GPS information attached to tweets is 1%-3% on average.

A study conducted on companies by Buddy Media [10] about usage of Twitter shows that tweets containing a photo link take attention as twice as the attention paid to tweets without a photo. Short tweets (less than 100 characters) are 17% more effective than the long tweets. People 86% more retweet the tweets including a link than the ones with no links. Tweets with hashtags are twice more effective than the ones with no hashtags. Followers retweet the tweets including the word "retweet" 12 times more.

A very detailed Twitter usage research has been conducted in 2013 by BuySellAds.com [2]. In this study; according to personal information like gender, homeland, race and income Twitter usage statistics were estimated. According to this study, 460K new members are joining Twitter every day. Comparing with 2011 data, there is a 182% increase in the users using mobile Twitter. 31% of male and 21% of female users use Twitter to get daily news. 24% of the users are using Twitter only to inform people about their location, 54% of the users write about their daily experiences and 40% use Twitter to send photos. If we look at the usage frequency, 24% of the users check their tweets more than once in a day. Media channels are seen as the most effective users. But this doesn't mean that people reach the news directly from these channels. It has been reported that only 15% of regular users get news from media accounts in Twitter. 71% of millions of tweets daily sent are not replied or retweeted. According to the research conducted by ComScore [2], the most frequently Twitter using country is Netherlands by 26.8%, followed by Japan 26.6% and Brazil 23.7%.

2.1.6. Twitter API

We preferred Twitter as social media system. And in order to collect data from Twitter we used Java library using Twitter API (Application Programming Interface where detailed information can be found in Section 4). By Twitter API data from Twitter can be collected in 3 different ways. These are classified as; REST API, Search API and Streaming API. We developed our prototype using Streaming API. Although there are 3 different data collecting API's, Search API and REST API can be basically classified as the same. The reason they were used to be classified differently in the documents is that search feature of Twitter used to be provided by a different company called Summize Inc. Although this company was acquired by Twitter later on, the search feature has not been added to Twitter codebase. The company was given a new name as Twitter Search [11]. For this reason, we can think Search API as a subheading of REST API.

2.1.6.1. REST API. Developers can reach timeline, status and user information data with REST API. Same kind of core Twitter data can be obtained by Search API. The main difference between REST and Search API is the usage limit. Two different limits can be applied in REST API. 15 requests per 15 minutes or 180 requests every 15 minutes. In other words, searching in REST API has been limited in 15 minutes window by 180 queries [12]. Rate limit in Search API, is not like REST API in which it is API requests per hour. It depends on the complexity of the query and frequency of the coming data. Rate limit depends on the IP of requesting client [13].

2.1.6.2. Search API. Search API allows you to get trend data based on a certain query word or location. As incoming data has a limit and we don't want spam in our data, Twitter filters returned answers. Returned data is 6-9 days Twitter recent data. Access is not possible to Tweets older than a certain date.

2.1.6.3. Streaming API. With Streaming API tweets can be delivered with certain query words. This flow is obtained near real time. There are 3 different types of

streaming endpoint. Public stream; public data is streamed, User streams; all data belonging to a certain user is streamed, Site streams; all data belonging to more than one user is streamed. By Streaming API only one long lived connection can be obtained. It aims to provide the connection stay on for a long time. The main difference between REST API and Streaming API is that Streaming API provides data near real time. Besides, with REST API you can request tweets in a certain area, tweets including a certain keyword and tweets based on user id and language type (AND query word logic). All of the filters can be applied. However there is OR logic in Streaming API. You can't apply all of the filters while importing data. If you want to make changes in your query words, you have to disconnect. Only a connection and a set of filters are permitted at a time. For more detailed information about Streaming API [14].

2.1.6.4. API parameters and format. Requests to REST and Streaming API returns in XML or JSON format. Whereas Search API supports ATOM and JSON formats. In our system, frequently used data by Streaming API are; user profile data (screen name, twitter userid, self entered profile location, timezone, follower/following count), tweet text, geo (coordinates of user if the tweet is geo-tagged).

2.1.6.5. HTTP Error status codes for Twitter API. Twitter API returns a response for every request. You can change the flow of the process depending on the returned response code. In Streaming API OAuth[1] only one long live connection can be provided by an access token. If you have a different OAuth token (belonging to a second Twitter account) you can secure a second connection from your IP address. Except for the near time tweet delivery, the main difference of Streaming API from REST API is that filters can only be applied once and can only be changed if disconnected. You can use another filter by using a second active connection. Some of the most frequently encountered status codes according to [15] are given below;

- *200 OK.* Query successful.
- *304 Not Modified.* There was no new data to return.

[1]OAuth: https://dev.twitter.com/docs/auth/oauth/faq

- *401 Unauthorized.* Authentication credentials were missing or incorrect.
- *403 Forbidden.* Access is not allowed. We received this error message in our system while reaching timeline of private profile user.
- *404 Not Found.* Requested URI is invalid. Most probably user does not exist.
- *429 Too Many Requests.* Applications rate limit having been exhausted. Please wait 15 minutes for request.

2.2. Text classification

In this section, we make an overview on text classification. This chapter includes text preprocessing, vector space model, feature selection, n-grams, collocation, and classification methods. Shortly, the aim of text classification is to predict the predetermined document classes in which our document exists [16].

2.2.1. Preprocessing

We preprocess all Tweets collected in this study before representing them as vector space model. For this thesis, preprocessing consists of a few stages. These stages are "stop word removal", "lemmatization" and "NER" (see Section 3).

2.2.1.1. Stop word removal. Stop word means the most frequently used words in a language. Our system handles English texts. For this reason, we eliminate frequently used words in English such as "the", "is", "at", "which", "and" and "on" from tweets (see Appendix A for full stop word list). Twitter messages are not written in a formal way as you can see in an ordinary news source and has its own jargon. For example the word ''RT" is a word frequently used in a sentence in Twitter. Besides the frequently used English words, we can also name the words frequently used in Twitter as "stop word". Abbreviations or repeating the letters in a word are frequently used in Twitter. For example the word"this" can be used as "ths" or as "thiiiiisss" by repeating a letter. Even thought the word seen in the example has been written differently, it is still a stop word and should be eliminated from the sentence. Changing the words with a

repeating letter into a word with no repeating can be provided by a simple algorithm. But more complex algorithms might be needed with the words where letters repeat in its original form. The word “glass” can be given as an example. We extended our stop word list staying suitable with the Twitter jargon. This extension has been obtained by adding the possible abbreviations of formal English words to the list. Of course this process, is not an exact solution for a system like Twitter where words are shortened by different combinations.

2.2.1.2. Lemmatization. Introducing the words into root form is a useful stage for indexing process. Words can be in different forms in sentences. For example “saw” is in the past tense whereas “see” is written in the present time. We call it “lemmatization” to change the words into their roots or dictionary forms. Stemming is the process of transforming the word into its root form taking out of the extensions from the word by the help of heuristic. Although forming the root form is possible by stemming, it is processed in a more proper way by lemmatization by the help of dictionary. For example the word “saw” may be turned into only the letter “s” after stemming. Lemmatization can change the same word “saw” into “see” or “saw” checking the part of the word in the sentence like if it is used as a verb or a noun [17]. As a more complex example the word “better” can be given. Stemmer will not be successful as it will not be able to find the root of the word by dictionary. But with lemmatization the word “good” will be turned as lemma. Although lemmatization is better at finding the root of a word, it may be slower than stemming. While using Stemmer, as precision decreases recall increases. Finding the correct root of the word is more important for our proposed model. For this reason, we preferred lemmatization as preprocessing and Stanford Core NLP library as Java implementation[2] has been used.

2.2.1.3. POS Tagging. POS (Part of Speech) We can determine the important features in a sentence (noun, adjective, verb) by tagging. As Twitter has a different corpus, POS taggers developed with newspaper corpus wouldn’t succeed. In this thesis, we

[2]Stanford Core NLP library: `http://nlp.stanford.edu/software/corenlp.shtml`

are using CMU POS Tagger NLP for Twitter.[3] Besides the parts of the sentences like noun, verb, adjective and adverb, we are tagging the items like hashtag, mention, URL and emoticon, special to Twitter by this library. According to Stanford POS tagger which is very popular[4] it has been evaluated that it is 25% more successful and has a success ratio of %90 in overall [18].

2.2.1.4. NER. We are trying to assign the terms in sentences into predetermined classes by NER (Named Entity Recognition). These classes can be; names of persons, organizations, locations, expressions of times. For this task, it is primarily important to build up a training set. Training set is built up by annotating the messages into a certain format (see Section 5.1 for sample annotation). To obtain a good named entity recognizer, many messages should be annotated. Although a NER study has been conducted over Twitter ([19,20]), there is no special training data prepared for emergency domain. For this reason, in our study we designed a training set containing approximately 1500 emergency tweets. Our training set has been separately annotated for location entities. For this process we used Apache OpenNLP.[5] For detailed theoretical information on Named Entity Recognition can be referred to this study [21, 22].

2.2.2. Vector Space Model

Vector Space Model is the representation of any document in a vector form. Each dimension of the vector is called a term. Document is made of words. Words, phrases and keywords can all be terms and represent the dimension of the vector. Terms mean differently in different applications. While representing a document as a vector, if the term exists in the document, the value of the related dimension is 1, if the term doesn't exist in the document this value is 0. If we choose the words in the document as term, then for example, dimension of the vector is equal to the number of the words in the document. Terms might have weights and algebraic processes can be done according to these weights. As an example for this, the most popular method to calculate the

[3]CMU Tweet NLP:http://www.ark.cs.cmu.edu/TweetNLP/
[4]Stanford POS tagger: http://nlp.stanford.edu/software/tagger.shtml
[5]Apache OpenNLP: http://opennlp.apache.org/

weight of the document is TF-IDF (Term Frequency - Inverse Document Frequency). After the vector representation of the document is done, we can use it as input for classification algorithms. While analyzing the tweets, we are using vector space model to determine the word frequency in our study.

2.2.3. Feature selection

Selection of the features is one of the important stages in text classification. In order to understand whether the tweets belong to a certain category, the characteristics of the tweets should be analyzed carefully. Document (tweets) will be turned into a vector which represent its own characteristics in the best way and contains the features expressing the differences. In our study, a good feature, should give close values for the compared same incident, and different values for different incident. Even though the incidents are different, feature vector should contain the same properties and it should be designed so that it helps to analyze the differences. By choosing good features, the dimension of the feature vector will decrease, classification accuracy will increase and it will require smaller computational requirements. Another advantage of feature selection is stated as its tendency to reduce overfitting [23].

2.2.4. N-grams

It is the listing of all combinations of n consecutive words in sentence. 1 length n-gram called unigram, 2 length bigram, 3 length trigram and n length called n-gram. It is generally used to predict the n^{th} word coming after the (n-1)st word composing the sentence. N grams are used to improve language detection, speech recognition, spelling correction or finding similar documents [24]. In this thesis, collocation is more important and analyzed in the subsection.

2.2.4.1. Collocation. Bi-gram or over n-gram sentences are not modeled randomly, they are modeled suitable with the grammatical structure of the language. An important application of bi-gram is collocation. Some bi-grams (or n-grams) in sentences

cannot be randomly seen together. “Turkish Republic”, “famous artist” examples can be given for bi-grams. In this study, collocations are given importance to analyze whether people use certain phrases frequently while sharing emergency situations in social media. For the determination of collocation, we used Log-Likelihood based Collocation Identification[6] method hosted in Apache Mahout[7] library.

2.2.5. Classification methods

Aim of classification is to automatically determine the class of an example. We can give two different approaches for text classification. These are supervised and unsupervised classification. In order to make a supervised classification, samples about the subject to be classified should be prepared by a human expert. Human expert prepares a set showing which class the samples belong to. The prepared sample set is called training set. By the help of training set, parameters and weights are calculated according to the methods. Basing on the calculated parameters, the class of the test sample is determined. For the unsupervised classification, we don’t need any training data. Our focus in this thesis is the supervised classifiers. In the following section, we will discuss about popular classifiers Naive Bayes and SVM.

2.2.5.1. Naive Bayes classifiers. Naive Bayes is one of the mostly used methods for text classification. Naive Bayes classifies with statistical methods depending on Bayes theorem. In this method, one of the most important assumptions is that, terms are independent and properties have the same weight. In the training stage, the frequency of terms in class of each document is determined. The probability of terms belonging to a class is calculated when the document to be classified has been received. This probability is calculated for each class in hand and document is classified in the class with the highest frequency. Theoretical information about Naive Bayes has been given in this study [25].

[6]Log-Likelihood based Collocation Identification: `https://cwiki.apache.org/MAHOUT/collocations.html`

[7]Apache Mahout: `https://cwiki.apache.org/MAHOUT/mahout-wiki.html`

2.2.5.2. Support Vector Machines. Support Vector Machines is one of the effective methods where we can apply linear classification technics to non linear data by the help of kernels. Kernels render the non linear data we have linearly separable in a multidimensional space. Our aim is to find the best hyper plane to classify the data into two different categories. Support vectors are parallel planes close to data to be separated. Hyper plane that will classify our group into two categories will exist at a point close to the groups and maximizing the margin in between two parallel planes. Detailed theoretical information about Support Vector Machines can be found in this study [26].

2.3. Linked data

Linked Data is huge amount of freely accessible machine readable data. Machine readable form of Wikipedia, which is one of the most frequently used encyclopedic sources is DBPedia and it is the heart of Linked Data. In this section, the aim of using Linked Data in this thesis and types of information to be obtained are explained.

2.3.1. About Linked data

Currently, information on internet are in the format readable by people and kept in databases at low levels. It is a problem to parse and use the information with different formats in database. Use of the information is beneficial only if information is transformed into a standard format. Generally when we analyze web, we already see some common standards. Addresses are determined with URL's and interfaces are prepared with HTML. We can call this application of common standard idea for sharing data among machines in order to make the data readable without any problem as Linked Data.

Common data format of Linked Data is RDF. RDF is composed of subject, predicate and object, called RDF triple [27]. We can think RDF triple as a graph model. In a RDF graph, nodes represent subject and object, whereas predicate represent the link between two nodes. Each resource is denoted by a URI (Uniform Resource Identifier).

While subject and predicates can be URI resource, objects can be URI resource or literal. Literals, text string (description), can be numerical values. URI's can only be represented as namespaces using abbreviation. For example: owl, dbpedia, skos. Predicates (edge) can be denoted as namespace:property. For example: While Geonames ontology is URI resource `http://www.geonames.org/ontology#`, "gn" can be used as namespace. While Property URI is `http://www.w3.org/2002/07/owl#PostalCode`, "PostalCode" prefix can be used. Consequently, as a short description"gn:PostalCode" can be used.

2.3.2. DBPedia

Wikipedia, Online Encyclopedia is the most popular site having the maximum amount of data among Wiki sites. This encyclopedia can be accessed in exactly 271 languages[8] and it is a huge source containing more than 10 million information. DBPedia project is the adapted form of gigantic Wikipedia source into RDF concept. Geographic information (countries, cities, seas, lakes), organizations (companies, hospitals, shopping centers), people (politicians, celebrities) are only few examples of machine readable templates existed in DBpedia. According to 2012 data[9] , English version of DBPedia includes 3.77 million "things", 764.000 person, 573.000 places, 192.000 organizations and more. It includes 20.8 million"things" in total and in 111 languages. Totally, it has got 1.89 billion of RDF triple. Out of this amount, 27 million belong to external RDF sets (GeoNames, Yago). Each resource can be accessed with HTTP. Response can be returned as RDF/XML, JSON, ATOM and HTML. For access to DBPedia and query we used DBPedia Spotlight[10] library. We extracted detailed information of entities we have recognized with this library from DBPedia. Detailed information about DBPedia Spotlight can be found in [28].

[8]stats as of 31.03.2013: `http://stats.wikimedia.org/EN/Sitemap.htm`

[9]DBPedia stats: http://blog.dbpedia.org/category/dataset-releases/

[10]DBPedia Spotlight: `https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki`

2.3.3. Using Linked data in case of emergency

The most important category of Linked Data for our project is location information. Location information can be extracted with NER from post. A good trained NER, is able to tag accurately the location information in different post coming from public stream. No matter how good NER has been trained, wrong tags will happen. To estimate the location of the event, this information should necessarily be expressed numerically (GPS coordinate). At this stage, a source like DBPedia provides us to access all kind of information regarding the location we have determined in written. Besides, in emergencies, people might explain the situation by using landmarks close to the area not directly the location. For example; there was a terrorist bombing at a bus-stop in Istanbul Etiler on 26.05.2011.[11] Bombing took place close to a shopping center (Akmerkez) which is popular in the region. While sharing this event in social media, instead of using the district's name (Etiler), people express the event as "Explosion at Akmerkez!". "Akmerkez" or in other words, organizations can be extracted from tweet with the help of a well trained NER that is also capable of recognizing organization names. As the exact location of this specific event is not specified in tweets, we can only make assumptions from profile/timezone information. Only the name of the city that the event took place can be determined by such an approach (considering a low ratio of GPS sharing). But organization detected by NER belongs to organization class in DBPedia. All kind of information regarding this organization exists in DBPedia:Place in machine readable format. By this way, the exact location of the event can be more accurately determined by DBPedia. Converting the location shared as a name into numerical form can be directly provided by Google Maps, Open Street Map. But people may always not share the name of the place directly in their messages. People can express the location in their messages in different ways. The most frequently used format of it is sharing organization names. At this point, DBpedia can play a great role in changing organizations into locations. If location is directly shared, determination of the type of location (city, town, state, country) or nearby locations can easily obtained by DBPedia.

[11] 26.05.2011 Akmerkez bombing: `http://www.hurriyetdailynews.com/default.aspx?pageid=438\&n=at-least-one-injured-in-istanbul-explosion-2011-05-26`

2.4. Non-Relational databases

We have to store the data in the most efficient way in order to be able to analyze a huge amount of data like Twitter. Today, many systems like Twitter, Facebook, LinkedIn, Amazon.com use NoSQL database systems to store and analyze huge volume of data. In this section, we will study what NoSQL database means and its types in detail. We will also talk about its advantages against traditional relational database systems. Although relational database systems are dominantly used by companies in the market, popularity of non relational database systems (a.k.a NoSQL databases) increases every day [29]. NoSQL techniques arise due to need of scaling heavy and frequent data read/writes, indexing large tables, serving high traffic web pages, stream media, handling inflexible/inconsistent attributes [30]. Advantages of NoSQL database systems are given in [29]. Briefly;

- *Schema-less.* Tables with no pre-defined schemas (may have partial or no schema). Tuples may differ by the number of fields they have. Allows forming classes of entities with complex internal structures (nested entities) meaning reduction of joins [31].
- *Elastic scaling.* Scaling out (distributing database as load increases) rather than scaling up (buying bigger servers as load increases). NoSQL databases are designed to expand to take advantage of new nodes added. Storage and server capacity can be added on the fly.
- *Big data handling.* Today many systems generate huge amount of data that cannot be managed easily and efficiently by relational databases.
- *Less management requirement.* NoSQL databases are designed to require less management; automatic repair, data distribution and simpler models. Those features lead to less management requirement.
- *Economics.* Clusters of cheap servers are necessary for NoSQL databases whereas relational databases need expensive servers and systems to accomplish same task. As the transaction cost decreases, NoSQL systems can process more data at lower prices.
- *Flexible data models.* It's generally difficult to change data model in relational

databases and needs careful management. System might need to be down or can be online with limited features. NoSQL databases are relaxed, no data model restrictions. Lack of schema and no necessity for joins [32]. Key value and document based databases allow storing any type of structures as elements. Even more strict models (i.e: BigData) allows you to add new columns to your design easily.

Of course not everything is perfect with NoSQL database systems. There are different challenges to be overcomed or solved. These challenges are shortly [29];

- *Maturity.* Relational databases are used for a long time, so they are rich in functionality and stable. However, NoSQL systems are generally "beta" and provide less functionality.
- *Support.* Technical support is crucial for enterprises and relational database vendors provide high level of technical support. NoSQL databases are generally open source projects and there might not be quality technical support for enterprises.
- *Development.* SQL is used for querying relational databases and even non-experts can write simple queries. However, in order to execute a query on NoSQL databases one needs programming expertise.
- *Management.* In theory NoSQL systems need less management. For now, installing and maintaining NoSQL databases needs effort.
- *Expertise.* Most of the problems in relational databases can generally be solved by people who are familiar with RDBMS concepts. NoSQL is pretty new and developers are also in learning mode.

NoSQL databases use few or no SQL queries to update data. As we have discussed above, they are designed for purposes in which relational database systems not designed for, such as scaling out and storing large amount of complex data (complex object would be difficult for relational databases to store in table) [32]. Relational databases handle small but frequent read/write operations well. However, they operate poorly for storing complex and large amount of data (i.e: indexing millions of data, retrieving objects,

streaming media files) [32]. NoSQL databases solve common problems for distributed databases by MapReduce and scaling. MapReduce is the model that NoSQL systems use for processing large amount of data. Two functions; map and reduce handle the processing of data across large scale [32]. It basically divides work into subworks for parallel processing. Complex queries can be divided into smaller parts, shared by other nodes and then answers can be merged for the result of the original query.

2.4.1. NoSQL models

There are several models for different needs in NoSQL databases for storing data.

- *KeyValue store.* Simplest NoSQL approach, data is stored with key (identifier) and data (value) like dictionary. Similar to two column table in relational databases, except value column can store multivalues [32]. i.e: Amazon SimpleDb, Scalaris, Redis.
- *Document-oriented database.* Structures data similar to document and does not use tables. Attributes of a schema can be dynamically added. Documents (tuples, records or rows in relational databases) may share common properties, but may also have different properties from each other [33]. i.e: MongoDb, CouchDb, Terrastore.
- *Column-oriented database.* Similar to relational databases, column store consists of tables containing addressable rows and rows consists of columns [34]. Although it might look similar to relational databases, column store has differences. Relational databases retrieve columns for a row in a table. It would be inexpensive to retrieve all the columns for a given row, however, it is expensive to retrieve all columns given a row in relational databases [33]. Column store NoSQL databases are optimized for retrieving multiple rows for a given column efficiently. Column values for multiple rows are stored in the same block, rather than storing multiple rows within the same block [34]. This is smart and faster because it's rare to query all the columns of a table. By this approach retrieving (updating) rows of the column to read will be faster. Each row in the table can have different set of columns (although rows may require to have predefined column groups), mean-

ing number of columns for values might change from row to row and columns can be added dynamically [34]. i.e: Google BigTable, HBase (basis on Hadoop), Cassandra, HyperTable.

- *Graph database.* Similar to graph structure (nodes, edges, properties), data is represented as objects(nodes) that relates with other objects. One possible example for usage might be about storing structures of social networking applications. Like document-oriented databases, graph database is not presented with tables, rows, etc. This means graph concept can be build by using different storage techniques [34]. i.e: Neo4j, AllegroGraph RDFStore.

In this study, we are using MongoDB, a document-oriented database system.[12] We chose a document-oriented structure because data we obtained from microblogging system, doesn't have a uniform size for every field. Each document (tweets) has its own characteristics. Considering that structure of Twitter is open to updates, new fields can be added to the document every day. Of course the mentioned properties can be provided in different NoSQL database systems. A document-oriented database inserts the whole document as a JSON object. But with a column-oriented database each individual column can be addressed and updated individually. Besides each column might have a separate timestamp or version. Generally with column oriented databases column values are in bytes but of course can be UTF8, text, timestamp or JSON. In the structure of a column-based database, we can't have the advantages we obtain from a document-oriented database system. While arbitrarily complex documents can be stored with a document-oriented database, one or two level dictionaries can be stored in column-oriented databases. Column-oriented structures are more suitable for updates. We can change a column even without reading the row in which the column exists. In our study, we experienced that the probability of updating fields of a document in such a system is low. Documents are processed, data formatted in a structured form and stored. In the light of this information, a document-oriented database has been preferred as it is highly scalable, efficient considering insert and suitable to format of Twitter data. Section 4 gives detailed information about our database structure.

[12]Mongo DB: `http://www.mongodb.org/`

3. MODEL

In this chapter, we propose a model for detecting incident related microblog posts, estimating incident location and notifying users from their mobile devices. At the highest level, our model called EmNotifier, can be viewed as follows:

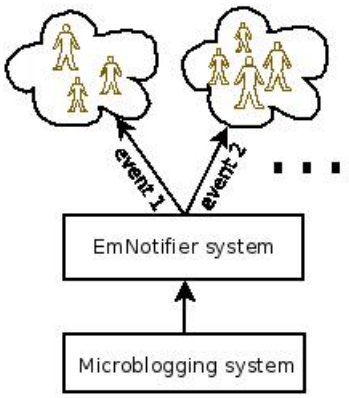


Figure 3.1. EmNotifier high level model.

Different sources can be used in order to determine incidents (see Section 3.1). Relatively, the most accessible and appropriate for text processing sources with the highest contribution ratio are microblogging systems. Of course because of microblogging systems nature, it is doubtful that we could gather the required information describing any emergency situation in detail. With our approach, we show that posts concerning different type of incidents can be extracted from microblog systems. If the location of the incident is determined, people around can be alerted. How soon the incident is determined depending on the type of the incident, people affected by this incident can be alerted in advance.

In this model, one of our main hypotheses is that microbloggers share about incident. Today, use of microblogs effectively shows us that existence possibility of this kind of contributions is high. Besides sharing, the most important information for incident is the information of location. As use of mobile applications increases, we see

that users are more willing to share their location.

Queries describing the incident should be developed in order to identify the posts users give information about the incident. Of course users don't only use the incident keywords for the incidents. For this reason, to find the best describing keywords for the incident and to separate them from the ones that are irrelevant with the incident are necessary. The main concern is to identify the location of the incident. Because of strategic reasons of using microblog, it is possible that user profile information may be incorrect. In the posts people share regarding the incident, we see that people begin to use location information in their posts. Many researches haven't used this valuable information. We, in our model, show how to extract the location information of the incident by using the users posts with the highest level.

For example, a user's profile location who posts"car fire at buyukdere ave., istanbul!" in his/her microblog can be a different city from "istanbul" which is written in his/her post. As we have sampled in Section 6 for determination of incident, location information mentioned in the post is highly possible the same with the exact incident location. A user may be not always updating his/her profile location, he/she may prefer not giving this information in his/her profile because of strategic or personal reasons. Besides, people are always in movement, change their places and they can't always update their profile. The idea is to use all possible location information given in a post for incident location estimation.

3.1. Incident data sources

We collect the data in real time with the help of social sensors in our model. Social sensors (User); can't share data instantaneously like electronic sensors and can't always be in active position under each circumstances. In our model, data concerning the emergency situation can be collected from different sources. Generally, social media systems can be a source to identify the incident. In short, characteristics of social media systems are;

- *Microblogging systems.* Microblogging systems are gigantic data sources in which every kind of information shared by the users. Sensors are called "citizen journalist". In our daily life, people momentarily share their feelings and experiences by microblogging systems. Data flow is fast. Trendy subjects change every minute. We can't rely on the accuracy of the information. This lack of trust for information is a problem but as the number of posts for similar perceptions is high, it can increase the credibility of the news. This is exactly crowdsourcing philosophy. As the post is not passed through the software language checks, possibility of spelling and grammar errors and abbreviations in words and sentences is high. Because of this feature, it decreases the rate of success in our present natural language processing systems.
- *Other social media systems.* Blogs, Wiki, Media Streaming systems fall into this category. Posts are relatively longer.

In order to reach definite and detailed data about incidents, online news sources or emergency related networks can be helpful. Briefly, features of these systems are;

- *News.* The most reliable sources are generally online news websites. Information is correct but it can't reach the updating speed of social media. At present, messengers follow up the social media more closely in order to learn the incidents more rapidly. In some social media systems posts are short. This is an advantage considering the text process and storage. In news websites, the shared news is always long. Text processing and data extracting are more complex. But as the shared data has been checked by the editors, the possibility of abbreviations and grammar errors is low.
- *Emergency communications network.* Emergency communication network[13] It is a network designed to provide a stabilized communication in emergency situations. Fire, police, and ambulance services use equipments like radio operators or pagers for communication. In case of an emergency, pager receives data like exact location address, level of importance and action code necessary to be taken. As technology develops every day, use of this kind of equipment in some areas

[13]ECN: `http://en.wikipedia.org/wiki/Emergency_communications_network`

are being replaced by cellular phones. For example, some fire departments[14] gave up using pagers and developed a system in which they receive the alert by their cellular phones. Of course this doesn't mean that pagers are completely replaced by cellular phones. Many fire departments still use tone/voice paging systems via certain frequencies. As a secondary alert text messaging is used. In places where cellular phone signals harm the medical equipments, like hospitals, less harmful systems are still being used. If information on this kind of networks is used, we can reach the detailed type and location data about the incident immediately.

If sources that can be used are investigated, we see that microblogging systems are more easily reached and we also see that they are systems in which more data is shared. Another advantage of social media systems, there is possibility of perceiving and damping the incident before it takes place. Other sources can only give exact information after the incident happens. As any incident is firstly received by emergency departments, these systems have the most detailed information about the incident. In our model, we are using the microblogging systems as the main source for incident detection. We show that people share in emergency situations (including local incidents) by using social media and we also show the level of location extraction in Section 6.

3.2. Incident characteristics and microblogger sharings

Primary aim of microblogging systems is not sharing data regarding the emergency situation. People share every kind of information (necessary and unnecessary) by these systems. In Microblogging systems some characteristics belonging to incidents are shared. Mostly used characteristics are;

- *Time of occurrence.* In order to send the notification as rapid as possible, the time of the incident is an important parameter. The response time for the incident should also be short. All process can be useless if a wrong time notice occurs. In microblogging posts, time information is given by items expressing time in the post (today, morning, etc.) or timestamp of post.

[14]Mobile emergency dispatch: `http://goo.gl/tN3a3`

- *Location.* For incidents with different scales, location information at different level is enough. For example; while for fire incidents location information as street and apartment number is required, city and country is enough for incidents like earthquakes. Analyzing microblogging systems, we see that location information is shared as given in Table 6.1.

As an addition to the list, scale of incident and status are shared by microbloggers for different incidents. It is the case that time of occurrence and location are enough to determine the incident and they are the basic characteristics of each incident. Incident characteristics mentioned below and shared by microbloggers may not be appropriate for every incident.

- *Scale of incident.* Some incidents affect a small area whereas some have an impact on a larger scale. i.e: local fire, forest fire, earthquake, flood, riot, hurricane.
- *Effected objects.* In order to give an accurate response to incidents type of the incident is sometimes important. Response can vary according to the type of the object affected. i.e: Fire (building, vehicle, forest, fire as a result of bombing), accident (vehicle).
- *Status.* Some incidents occur momentarily and some continue for days. i.e: Fire (forest, vehicle), earthquake (happens in a few minutes and finishes).

High level representation of our model is given in Figure 3.1. At highest level our purpose is to send emergency notification to people across the globe. Clouds represents different locations. Primarily people inside the cloud represent the ones that live or passing nearby the emergency area and secondarily people that are interested with the region.

EmNotifier proposes a model for incident detection using microblogs and mobile notification by;

- Identifying a list of incident related query words.
- Fetching posts by using near real time query word filtered public stream connection from microblogging systems.
- Classifying posts of microbloggers based on incident types as related or not related to incident.
- Extracting location information from microblog posts.
- Aggregate posts that are related to same incident by calculating the geographical distance between posts, posted within close time intervals.
- Send interested mobile users notification.

Primary purpose of our model is to detect the presence of an incident and its location. Before going deeply about the model, we introduce data types, functions and constants. Data types we have used describing the model are; PrimitiveDT (Table 3.1), MicroblogDT (Table 3.2), MicroblogProcessingDT (Table 3.3), EmNotifier ProcessingDT (Table 3.4) and constants (Table 3.5).

Table 3.1. PrimitiveDT: Primitive data types for handling microblogs.

Type Name	**Description**
Bool	a boolean value
Integer	an integer value
Long	a long value
Double	a double value
Text	a sequence of characters including white spaces

The terms set, bag, list are frequently used in the context of text processing. Our convention for describing them can be given as; Sett_s, List_l and Bagg_b. For instance a query consists of a set of word Word_s. In our model users are constantly in motion.

Incidents can be fixed to a place or propagate. Scale and duration of incidents might change according to incident type.

First step is to fetch posts from microblogging system. Function *StreamReader* (See Figure 3.3) takes a query that best describes an incident, provides a single live connection with the microblogging system and fetches related posts. For instance "Earthquake" incident related query words can be "earthquake, temblor, ground shaking, quake". Synonyms of incident describing words can be used to increase the chance to fetch more incident related posts. For instance for "fire" keyword "burn, burn down" are some synonyms.

Microblogging system streams unstructured posts. In order to use unstructured posts in the system, a processing takes place to convert them to feature enriched posts. Feature enriched posts have been used for all steps in our model.

Definition 3.1. Unstructured post *is raw microblog entry fetched from a microblogging system (see Figure 3.2).*

Table 3.2. MicroblogDT: Data types of a microblogging system.

imports PrimitiveDT		
Type Name	**Specification**	**Notes**
Post	Text	The text of the content in a microblog post.
MBlogEntry	< ScreenName, Post, Timestamp, Timezone, GPS, ProfileLocation>	A microblog entry consisting of a name and post content.
ScreenName	@[A-Za-z0-9_]+	Microblog user name.
Token	Word | Tag	Types of token in a post.
Tag	Hashtag | Punctuation | URL | Emoticon | Mention	Types of tags in a post.
Emoticon	$[? :: \|; \| =)(? : -)?(? :)\|D\|P]$	All emoticons.
Punctuation	[.!?−]	All punctuations.
Hashtag	#[A-Za-z0-9_]+	A microblog tag.
Mention	ScreenName	A reference to a microblogger.
ProfileLocation	Text	Manually entered microblogger location.
Timezone	Text	List selected microblogger timezone.
MicroBlogSystem	MBlogEntry_l	Microblog system consisting of microblog entries

Table 3.2. MicroblogDT: Data types of a microblogging system (cont.).

imports PrimitiveDT		
Type Name	**Specification**	**Notes**
Word	[A-Za-z0-9_]+	a sequence of characters delimited by white space.
URL	[(http:[/][/] |www.)([a-z] |[A-Z] |[0-9] |[/.] |[com])*)]	uniform resource locator as defined by W3C
GPS	<Double, Double>	Point consisting of longitude and latitude
Timestamp	Long	Epoch time

Table 3.3. MicroblogProcessingDT: Data types for processing microblogs.

imports MicroblogDT	
Type Name	**Description**
Query	Word_s
TaggedToken	<Index,Word,Tag>
Index	Int
Frequency	Int
Classifier	"positive" | "negative"
TokenFrequency	<Frequency,Token>
FeatureEnrichedPost	<MBlogEntry, FeatureVector, Classifier,Location_l>
FeatureEnrichedPosts	$\text{FeatureEnrichedPost}_l$
FeatureVector	<Int,TokenFrequency, TaggedToken,TaggedToken>
Location	<Street, Town, City, State or Province, Country, County or District, GPS>

Table 3.4. EmNotifier ProcessingDT: Data types for EmNotifier.

imports MicroProcessingDT	
Type Name	**Description**
IncidentType	Word describing incident
Incidents	IncidentEntry_l
IncidentEntry	<Timestamp, IncidentType, Location_l, $\text{FeatureEnrichedPost}_l$ >
CurrentLocation	Current Location of Mobile User
MobileUserID	Mobile Device ID
MobileUsers	MobileUser_l
MobileUser	<CurrentLocation, MobileUserID>

Table 3.5. Model Constants.

Constant	**Description**
NotifyDistanceThreshold	Distance measure for triggering notification to MobileUser for an incident. Might change corresponding to scale of incident.
NotifyTimeThreshold	Measure specifying the duration that notification be active. Might change corresponding to duration of incident.
TimeThreshold	Time metric used for aggregating FeatureEnrichedPost on a single incident.
DistanceThreshold	Distance metric used for aggregating FeatureEnrichedPost on a single incident.
SimilarityThreshold	Threshold for the similarity method used (ex: Jaccard similarity).
MBlogstream	Provides a connection to any microblogging system.

Table 3.6. Text processing related functions.

Function	**Description**
TokenizePost(Post) : Token_b	Tokenizes a given post into Token_b.
TagPost(Token_b) : TaggedToken_s	Tags every token in Token_b.
StopWordEliminate(TaggedToken_s) : TaggedToken_s	Eliminates stop words from a given TaggedToken_s using a stop word list (see Appendix A).
LemmatizeText(TaggedToken_s) : TaggedToken_s	Determines the lemma for every token in a given TaggedToken_s.
Count(TaggedToken_s) : Int	Counts the elements in a given TaggedToken_s.
PostWordContent(TaggedToken_s) : TokenFrequency_s	Determines the occurrence of tokens given TaggedToken_s.
BoundaryWords(TaggedToken_s,Word_s) : TaggedToken_s	Determines the boundary words for every Word.
DetectLocationNER(TaggedToken_s) : TaggedToken_s	Determines location entities in a given TaggedToken_s using manually annotated data and supervised learning.

Table 3.7. FeatureEnrichedPost related functions.

Function	**Description**
AddToFeatureEnrichedPosts(FeatureEnrichedPost, FeatureEnrichedPosts) : void	Inserts FeatureEnrichedPost into list of FeatureEnrichedPosts.
GetAllFeatureEnrichedPosts(void) : FeatureEnrichedPosts	Returns all FeatureEnrichedPost in FeatureEnrichedPosts.
GetClassfiedAs(FeatureEnrichedPost) : Classifier	Returns Classifier describing which of the predetermined class type FeatureEnrichedPost belongs to.
GetPostProfileLocation(FeatureEnrichedPost) : Word_s	Returns user profile information in MBlogEntry.
GetPostLocationEntities(FeatureEnrichedPost) : Word_s	Returns a list of location entities detected by NER in post.
GetPostGPSLocation(FeatureEnrichedPost) : Location	Returns GPS information attached to microbloggers contribution.
CreateNewFeatureEnrichedPostList() : FeatureEnrichedPosts	Initiates a new list of FeatureEnrichedPosts.

Table 3.8. Interacting with Microblogging system related functions.

Function	**Description**
StreamReader(Query) : void	Establishes a single long-live connection with microblogging system for fetching Query based Unstructured Message$_l$.
Connect(MBlogstream) : MicroBlogSystem	Provides a connection to given MicroBlogSystem.
GetMBlogPostText(MBlogEntry) : Post	Returns contributors post.
GetMBlogEntries(MicroBlogSystem,Query) : MBlogEntry$_l$	Filters MicroBlogSystem based on Query.

Table 3.9. Classification related functions.

Function	**Description**
PrepareFeatureVector(Post) : FeatureVector	Prepares selected features as an input to classification algorithm, arranges features to a feature vector.
ClassifyUsingSupervisedLearning(FeatureVector) : Word	Classifies a given FeatureVector into one of the predetermined classes using machine learning methods.

Table 3.10. Incident detection related functions.

Function	**Description**
IncidentDetection(IncidentType) : void	Main function for detecting incidents.
IncidentCandidateCheck(IncidentEntry) : void	Identifies if a FeatureEnrichedPost belongs to an ongoing incident or starter for new incident.
GetLocationFromIncidentEntry(IncidentEntry) : Location$_l$	Returns a possible Location$_l$ that incident might belong to.
CalculateDistance(Location$_l$,Location$_l$) : Double	Calculates distance for all possible matches between two Location$_l$. Returns the minimum distance calculated.
GetTimestampFromIncidentEntry(IncidentEntry) : Timestamp	Returns timestamp of the first FeatureEnriched-Post that belongs to an incident.
NewIncident(Incidents,IncidentEntry) : void	Inserts a new IncidentEntry to Incidents.

Table 3.10. Incident detection related functions (cont.).

Function	**Description**
UpdateIncident(IncidentEntry,IncidentEntry) : void	Updates an IncidentEntry with a new incident candidate. Estimates the location of incident by aggregating all incident entries.
GetAllIncidentsByPriority() : IncidentEntry$_l$	It is possible that a priority exists between incidents. For instance, a local fire incident might have higher priority than earthquake incident for a local.

Table 3.11. Location extraction related functions.

Function	**Description**
ExtractLocationFromFep(FeatureEnrichedPost) : Location_l	Main function for determining possible locations from a FeatureEnrichedPost.
CalculateDistance(Location_l,Location_l) : Double	Calculates the distance between two Location in distance units.
ExtractLocationFromPost(FeatureEnrichedPost) : Location_l	Determines possible locations from a post.
MatchLocations(Location_l,Location_l) : Location_l	Matches all possible combinations of Location for two Location_l based on Country field.
LocationsFromTimezone(FeatureEnrichedPost) : Location_l	Determines all possible Country&City tuples for a given timezone.
ExtractLocationFromProfile(FeatureEnrichedPost) : Location_l	Extracts profile location by comparing other location information denoted in FeatureEnrichedPost.
SemanticSearch(Word_s) : Location_l	Uses semantic sources to get valuable information (Town, City, State or Province, Country, County or District, GPS) given a location.

Table 3.12. Notification related functions.

Function	**Description**
Notifier() : void	Determines the MobileUser$_l$ to be notified, traverses all the incidents for triggering notification according to incident priority.
SendPushNotification(MobileUser) : void	Sends push notification to MobileUser's mobile device.
GetAllMobileUsers() : MobileUsers	Returns all the MobileUsers.
CurrentLocationFromMobileUser(MobileUser) : Location$_l$	Returns the location of MobileUser.

Definition 3.2. Feature enriched post *enriches the unstructured post by extracting statistical information as the count of emoticons, hashtags, URLs, tokens, mentions, location entities. Additionally feature enriched post contains the information on which predetermined class the post belongs.*

It is highly possible that microblogging system pushes high number of posts that are completely unrelated to an incident candidate (ex: song lyrics). Posts that are related to an incident should be filtered from unrelated ones. It is not guaranteed to find posts about every incident. Similar to failure of electronic sensors, human sensors could be sleeping or might not contribute to microblogging system if they are in a life threatening emergency situation.

3.3. Classification

While fetching posts from a microblogging system, despite query words that best expresses an incident are used, each MBlogEntry will not be associated with an incident. Every unstructured post have been assigned to a predetermined class while becoming feature enriched post. These are"positive" and"negative" classes. Several examples of incident related and unrelated posts are given in Section 6.

Definition 3.3. Positive feature enriched post *is related to an incident candidate.*

Definition 3.4. Negative feature enriched post *is completely unrelated to any incident or the desired incident type.*

Regardless of supervised machine learning method used, the most important phase of classification is feature selection. Feature enriched post should be an input to system that consists of features that maximizes the difference between classes. Accuracy for the incident detection is highly correlated with the features selected (see Table 5.4, 5.5).

Contribution characteristics for incident related and unrelated posts are differ-

“completed_in”:0.01,
“max_id”:338020701250404352,
“max_id_str”:“338020701250404352”,
“page”:1,
“query”:“from%3AHakanAnit”,
“results”:
{
“created_at”:“Fri, 24 May 2013 19:56:29 +0000”,
“from_user”:“HakanAnit”,
“from_user_id”:1324233,
“from_user_id_str”:“1324233”,
“from_user_name”:“Hakan Anit”,
“geo”:null,
“d”:338020701250404352,
“id_str”:“338020701250404352”,
“iso_language_code”:“en”,
“text”: jquery: Just in time, jQuery 1.10.0 and 2.0.1 are out!
}

Figure 3.2. An example of raw microblog post.

Require: *query*, Query words related to an incident
MicroBlogSystem *mblogSystem* ← Connect(*MBLOG_STREAM*)
MBlogEntry$_l$ *mblogEntries* ← GetMBlogEntries(*myblogSystem*,*query*)
MBlogEntry *mbe*
FeatureVector *featureVector*
Classifier *classifiedAs*
FeatureEnrichedPost *featureEnrichedPost*
FeatureEnrichedPosts *FeatureEnrichedPosts* ← CreateNewFeatureEnrichedPostList()
for each *mbe* in *mblogEntries* **do**
post ← GetMBlogPostText(*mbe*)
featureVector ← PrepareFeatureVector(*post*)
classifiedAs ← ClassifyUsingSupervisedLearning(*featureVector*)
featureEnrichedPost ← <*mbe*,*featureVector*,*classifiedAs*,∈>
AddToFeatureEnrichedPosts(*featureEnrichedPosts*,*featureEnrichedPost*)
end for

Figure 3.3. StreamReader algorithm to fetch query based posts from microblog system.

Require: *post*, microblog post of type Post

Token$_b$ *rawTokenizedPost*

TaggedToken$_s$ *taggedPost*,*taggedPostElim*,*lemmatizedText*,*taggedPostElim*,*boundaryWords*

Int *stopCountPost*

TokenFrequency$_s$ *tagFreqSet*

FeatureVector *featureVector*

Word$_s$ *boundaryTags*

rawTokenizedPost ← TokenizePost(*post*)

taggedPost ← TagPost(*rawTokenizedPost*)

taggedPostElim ← StopWordEliminate(*taggedPost*)

stopCountPost ←Count(*taggedPostElim*)

tagFreqSet ←PostWordContent(*taggedPostElim*)

lemmatizedText ←LemmatizeText(*taggedPostElim*)

taggedLemmatizedText ←DetectLocationNER(*lemmatizedText*)

boundaryTags ← <*query*,"hashtag","emoticon","punctuation","url","mention","location">

boundaryWords ←BoundaryWords(*taggedLemmatizedText*, *boundaryTags*)

featureVector ← <*stopCountPost*,*tagFreqSet*,*taggedLemmatizedText*,*boundaryWords*>

return *featureVector*

Figure 3.4. PrepareFeatureVector algorithm to prepare feature vector for classification.

Raw contribution: BigTex just BURNED DOWN. I was there, my kids watched it. So sad :-(#Texas @BigTex http://t.co/yvnGnvEF
Processed contribution: BigTex burn EMOTICON LOCATION MENTION URL

Figure 3.5. An example of raw and processed microblog post text.

ent. Additionally contribution characteristics for different types of incidents might vary. Basic statistical characteristics for several types of contributions are denoted in Table 6.1. Based on these characteristics a feature vector have been prepared for classification (see Figure 3.4). PrepareFeatureVector function takes a Post as input and by determining basic statistical information of post generates a FeatureVector. An example of raw post and processed contribution is given in Figure 3.5. A post is tokenized using TokenizePost function. Tokens for the given example would be; Token_b tokens $\leftarrow$ {"BigTex", "just","BURNED", "DOWN.", "I", "was", "there,", "my", "kids,","watched", "it.", "So", "sad", ":-(", '#Texas","@BigTex", "http://t.co/yvnGnvEF"}.

Function StopWordEliminate eliminates frequently used English words using a predefined vocabulary given a Token_b. TagPost function tags the entities given a Token_b and the output is TaggedToken_s. For the given example the output of TagPost will be; TaggedToken_s tokens $\leftarrow$ {<0,"BigTex",∈>,<1,"just",∈>,<2,"BURNED",∈>, <3, "DOWN.",∈>,<4,"I",∈>, <5,"was",∈>,<6,"there",∈>,<7,"my",∈>,<8,"kids",∈>, <9,"watched",∈>, <10,"it.",∈>, <11,"So",∈>, <12,"sad",∈>, <13,":-(", "EMOTICON">, <14,"#Texas","HASHTAG">,<15," BigTex","MENTION">, <16,"`http://t.co/yvnGnvEF`","URL">}.

Function PostWordContent counts the number of special items in a post and return TokenFrequency. For the example it is TokenFrequency frequency $\leftarrow$ {<1,"HASHTAG">,<1,"URL">,<1,"MENTION">,<1,"EMOTICON">}.

LemmatizeText function determines the lemma of a given TaggedToken$_s$. It does not determine lemma's of Tag. For the example the output is; TaggedToken$_s$ lemmatizedtokens ← {<0,"BigTex",∈>,<1,"just",∈>,<2,"BURN",∈>,<3,"DOW.",∈>, <4,"I",∈>,<5,"was",∈>,<6,"there",∈>,<7,"my",∈>,<8,"kid",∈>, <9,"watch",∈>, <10,"it.",∈>,<11,"So",∈>,<12,"sad",∈>,<13,":-(","EMOTICON">, <14, "#Texas", "HASHTAG">, <15," BigTex","MENTION">, <16,"http://t.co/yvnGnvEF","URL">}.

DetectLocationNER detects location entities in a given Token$_b$. It uses a dataset of location annotated microblog posts for training (see Section 5.1 for sample annotation). PrepareFeatureVector function uses determined basic statistical features and tags for preparation of classification. Output of PrepareFeatureVector function is used in ClassifyUsingSupervisedLearning function. Index of location describing entity in post, boundary words and stop word count in a post are the most important features in our model.

3.4. Incident detection

Feature enriched posts that are classified as negative have not been used for incident detection. MBlogEntry contains manually entered user profile location, timezone and if activated by the application, GPS position of microblogger. IncidentDetection function (see Figure 3.6), determines the location for every positive FeatureEnrichedPost in FeatureEnrichedPosts according to IncidentType. Extracted location can be multiple, Location$_l$ meaning there is a need for location disambiguation. This generally occurs when the MBlogEntry only contains of timezone information, or place names that are not unique. For incident aggregation our model basis on the idea that, in a certain time and location there can only happen a single incident of same type.

IncidentCandidateCheck function (see Figure 3.7) identifies if a FeatureEnriched-Post belongs to an ongoing incident or is a starter for new incident. For every Incident in Incidents it calculates the distance between the location of incident and incident candidate. If the calculated distance and timestamp between incidents conforms with

Require: *incidentType*, type of incident
FeatureEnrichedPost *fep*
FeatureEnrichedPosts *fepList* ← GetHighPriorityFep()
Location$_l$ *locationFep* //FeatureEnrichedPost might contain multiple locations.
IncidentEntry *incidentCandidate*
for each *fep* in *fepList* and GetClassifiedAs(*fep*)==“positive” **do**
locationFep ← ExtractLocationFromFep(*fep*)
incidentCandidate ← <GetTimestamp(*fep*),*locationFep*,*incidentType*,*fep*>
IncidentCandidateCheck(*incidentCandidate*)
end for

Figure 3.6. IncidentDetection algorithm to detect incidents.

DistanceThreshold and TimeThreshold than incidents are aggregated based on UpdateIncident function. In case there are multiple locations for single Location$_l$ input in CalculateDistance function, it calculates the distance between all possible mappings for both of the Location$_l$ inputs. The result is the minimum calculated distance. This way MBlogEntry that contains only timezone information can be assigned to an incident. UpdateIncident function determines the methodology to target incident center. A basic approach is to take the average of contribution locations as the center of incident.

3.5. Location extraction

All the location indicators in FeatureEnrichedPost are used for a decision to be made about assigning FeatureEnrichedPost to location. Location indicators in FeatureEnrichedPost are depicted in Table 3.13. ExtractLocationFromFep function (see Figure 3.8) takes FeatureEnrichedPost as an input, uses all possible location indicators in FeatureEnrichedPost and decides on possible location(s) of FeatureEnrichedPost. We have inspected microblog posts to determine a priority for using location indicators in FeatureEnrichedPost to find out the highest possible location of incident. In order to correctly estimate the location of incident, we would like to find contributors

```
Require: incidentCandidate, IncidentEntry
  Boolean boolIncidentAdded ← false
  Incidents incidents ← GetAllIncidents()
  Timestamp tDiff
  Location_l locationIncident, locationCandidate
  Double distance
  IncidentEntry incident
  for each incident in incidents and boolIncidentAdded==false do
    locationIncident ← GetLocationFromIncident(incident)
    locationCandidate ← GetLocationFromIncident(incidentCandidate)
    distance ← CalculateDistance(locationIncident,locationCandidate)
    tDiff ← GetTimestampFromIncident(incidentCandidate) -
    GetTimestampFromIncident(incident)
    if distance ≤ DistanceThreshold && tDiff ≤ TimeThreshold then
        UpdateIncident(incident,incidentCandidate)
        boolIncidentAdded ← true
    end if
  end for
  if boolIncidentAdded==false then
    NewIncident(incidents,incidentCandidate)
  end if
```

Figure 3.7. IncidentCandidateCheck algorithm to determine if an incident candidate belongs to new or ongoing incident.

that report inside from the incident (see Figure 6.2, 6.1). We have prioritized the use of location information in contributor sharings as; in post location indicators, manually entered profile location or if available attached GPS and timezone respectively.

ExtractLocationFromFep is the main function for location extraction from FeatureEnrichedPost. Based upon the priorities defined, ExtractLocationFromFep function estimates a list of possible location(s) for a given FeatureEnrichedPost. It uses all the location indicators in a FeatureEnrichedPost for making a decision about the location of incident. If the location indicator is a GPS point, then the output is a single location. In case location indicator is timezone, all the cities that belong to it are possible candidate locations.

According to our model, if the contributors sharing contains "in post" location indicator, it is the most valuable location information type. In addition to "in post" location, as can be seen in Table 6.1 different types of location indicators are also used for incident location estimation.

Table 3.13. Types of location related elements in FeatureEnrichedPost.

Type Name	**Description**
Profile Location	User entered profile location in Text
Timezone	User selected timezone from a list in Text
GPS	Optionally attached GPSinformation
Post	As discussed in Section 6 users share location information in their posts

Place names are not unique and might belong to different locations. For instance, place name "Paris" belongs to at least 5 different locations (see Figure 3.9). That is determined by using semantic sources. One of the "Paris" belongs to country France as city and the others are cities in different counties of United States (Paris in Logan county, Lamar county, Bourbon county, Monroe county). It is possible to use semantic

Require: *fep*, FeatureEnrichedPost
Location$_l$ *locationFep*
Location$_l$ *locationProfile* ← ExtractLocationFromProfile(*fep*)
Location$_l$ *locationPost* ← ExtractLocationFromPost(*fep*)
Location$_l$ *locationTZ* ← AllLocationsFromTimezone(*fep*)
Location *locationGPS* ← GetGPS(*fep*)
Location *locationFromPostAndGPS* ← MatchLocations(*locationPost*,*locationGPS*)
Location$_l$ *locationFromPostAndProfile* ← MatchLocations(*locationPost*,*locationProfile*)
Location$_l$ *locationFromPostAndTZ* ← MatchLocations(*locationPost*,*locationTZ*)
if *locationFromPostAndGPS* $\neq \in$ **then**
locationFep ← *locationFromPostAndGPS*
else if *locationFromPostAndProfile* $\neq \in$ **then**
locationFep ← *locationFromPostAndProfile*
else if *locationFromPostAndTZ* $\neq \in$ **then**
locationFep← *locationFromPostAndTZ*
else if *locationPost* $\neq \in$ **then**
locationFep ← *locationPost*
else if *locationGPS* $\neq \in$ **then**
locationFep ← *locationGPS*
else if *locationFromProfile* $\neq \in$ **then**
locationFep ← *locationFromProfile*
else if *locationFromTZ* $\neq \in$ **then**
locationFep ← *locationFromTZ*
end if
return *locationFep*

Figure 3.8. ExtractLocationFromFep algorithm to extract location from feature enriched post.

Location$_l$ locations ← <∈,∈, “Paris”,“Ile de France”,
“France”, 48.85,2.35>,
<∈,∈, “Paris”,“Arizona”, “USA”,“Logan”, 35.29,-93.73> ,
<∈,∈, “Paris”,“Texas”, “USA”, “Lamar”, 33.661,-95.5>,
<∈,∈, “Paris”,“KY”, “USA”, “Bourbon”, 38.209,-84.252>,
<∈,∈, “Paris”,“MO”, “USA”, “Monroe”, 39.481,-92.01>

Figure 3.9. An example for possible Location$_l$,“Paris”.

source to find which one of them is popular.

Algorithm MatchLocations takes two Location$_l$ as input and returns a list of possible location matches for the given input. For example given a list of two locations defined as Location$_l$ T and Location$_l$ P, where T and P stands for “in post location” and “profile location” respectively, function MatchLocations will return a list of possible locations that match according to “country” field, L.

<∈,∈, “Fairfield”,“California”, “USA”,“Solano”, 38.25,-122.04>
<∈,∈, “Fairfield”,“Fairfield”, “USA”,“Connecticut”, 41.14,-73.26>
} T

<∈,∈, “Fairfield”,“Iowa”, “USA”,“Jefferson”, 41.01,-91.96>
<∈,∈, “Fairfield”,“NSW”, “Australia”,∈, -33.87,150.95>
} P

<∈,∈, “Fairfield”,“California”, “USA”,“Solano”, 38.25,-122.04>
<∈,∈,“Fairfield”,“Fairfield”, “USA”,“Connecticut”, 41.14,-73.26>
<∈,∈, “Fairfield”,“Iowa”, “USA”,“Jefferson”, 41.01,-91.96>.
} L

```
Require: locationList1, list of possible locations (Location_l)
Require: locationList2, list of possible locations (Location_l)
  Location_l locationList, locationIntersectionList
  Location locationTemp1, locationTemp2
  Word country1, country2
  for each locationTemp1 in locationList1 do
    country1 ← getCountryFromLocation(locationTemp1)
    for each locationTemp2 in locationList2
    && locationIntersectionList.contains(locationTemp2)==false do
        country2 ← getCountryFromLocation(locationTemp2)
        if country1==country2 then
            locationIntersectionList.add(locationTemp2)
            if locationIntersection.contains(locationTemp1)==false then
                locationIntersection.add(locationTemp1)
            end if
        end if
    end for
  end for
  return locationIntersectionList
```

Figure 3.10. MatchLocations algorithm to match all possible combinations of Location.

Definition 3.5. Sensitivity *is defined as precision of the location indicator. Country information is low sensitive whereas GPS is high sensitive.*

SemanticSearch function uses semantic sources to get all possible place information for a given Word$_s$. We are using semantic sources because they provide additional information about a location, like neigbourhoods, abbreviations, popular cities, GPS. The other reason is contributors might refer to a nearby landmark rather than directly mentioning a city, country name. An example for a list of possible locations that are returned by semantic sources for word "Paris" is given in Figure 3.9.

ExtractLocationFromProfile function takes FeatureEnrichedPost as an input and returns a list of possible locations that are extracted from user profile (Word$_s$). User profile location is compared with timezone and GPS if available, for limiting the list of possible locations.

Require: *fep*, FeatureEnrichedPost
  Word$_s$ *textProfileLocation* ← GetProfileLocation(*fep*)
  Location$_l$ *locationProfile* ← SemanticSearch(*textProfileLocation*)
  Location$_l$ *locationTZ* ← AllLocationsFromTimezone(*fep*)
  Location *locationGPS* ← GetGPS(*fep*)
  //Purpose is to limit the search space of locationProfile using possible other valuable information
  Location *locationFromProfileAndGPS* ← MatchLocation(*locationProfile*,*locationGPS*)
  Location$_l$ *locationFromProfileAndTZ* ← MatchLocation(*locationProfile*,*locationTZ*)
  if *locationFromProfileAndGPS* $\neq \in$ **then**
    locationProfile ← *locationFromProfileAndGPS*
  else if *locationFromProfileAndTZ* $\neq \in$ **then**
    locationProfile ← *locationFromProfileAndTZ*
  end if
  return *locationProfile*

Figure 3.11. ExtractLocationFromProfile algorithm to extract profile location from microblog user profile.

ExtractLocationFromPost function uses the location entities "in post" detected by named entity recognizer function DetectLocationNER and user profile location for extracting possible locations from post. Similarity function takes two Word_s and calculates the text similarity. In microblogging environment it is highly possible that contributors misspell words. A specific example is given in Table 3.14 where a contributor is mentioning about a location and place where fire incident took place. From trustworthy sources we have found that real incident location is in "Fairfield, California". User profile location is given as "Fairfield, California". Although the contributor mentions about the same incident occurring in "Fairfield", he/she misspells the word as "Farfield". It is highly possible that a place named "Farfield" exists, and it is.

Table 3.14. An Example for misspelling in microblogging systems.

Post	**Profile location**
#Pepperbellys on fire in downtown **Farfield**	**Fairfield**, California

Taking the similarity of the location describing entity "in post" and "profile location", a chance might be given to post that the incident took place in Farfield and Fairfield. This is done by determining the profile location as all possible locations mentioning Fairfield and Farfield using semantic sources. Of course there is a chance that there is another fire incident in Farfield. The reason we have not discarded Farfield is because there might be an incident going on there, that was mentioned by previous contributions. Based upon crowdsourcing philisophy, calculating the distance between previous contribution locations and the location of FeatureEnrichedPost that conform to distance and time threshold, function IncidentCandidateCheck will determine the possible location of the incident. If the previous contributions were from Fairfield, FeatureEnrichedPost will be attached to that incident.

3.6. Notification

In our model, every user that we are going to notify constantly move and their location is pushed to system periodically. Incidents can be fixed to a place or move based

Require: *fep*, FeatureEnrichedPost
  Location$_l$ *locationPost*
  Word$_s$ *textPostLocation*← GetLocationEntitiesFromPost(*fep*)
  Word$_s$ *textProfileLocation*← GetProfileLocationFromFep(*fep*)
  if Similarity(*textPostLocation*,*textProfileLocation*) ≥ *SIMILARITY_THRESHOLD* **then**
    locationPost.Add(SemanticSearch(*textProfileLocation*))
    locationPost.Add(SemanticSearch(*textPostLocation*))
  else
    locationPost.Add(SemanticSearch(*textPostLocation*))
  end if
  return *locationPost*

Figure 3.12. ExtractLocationFromPost algorithm to extract location from post.

on their types. Figure 3.13 denotes notification process for incidents detected. Notifier function calculates the distance between the current location of user and estimated location of incident. According to NotifyDistanceThreshold and NotifyTimeThreshold, users that are possibly be affected by the incident will be notified. A user might travel to a region or get out from a region where an incident is happening. Once the user is in affected region specified by the NotifyDistanceThreshold, they get a notification. NotifyDistanceThreshold is defined according to scale of incidents. Status of incident might be updated anytime by contributors. For instance, fire incident might propagate to another region by the effect of wind. Until the region is safe, users will be notified by the latest update. NotifyTimeThreshold specifies the duration in which notification will be active. For different types of incidents we have to change the values of NotifyDistanceThreshold and NotifyTimeThreshold.

```
IncidentEntry incident
Incidents incidents ← GetIncidentsByPriority()
MobileUser mu
MobileUsers mobileUsers ← GetMobileUsers()
Double distance
Timestamp timestampDifference
while true do
   for each incident in incidents do
      for each mu in mobileUsers do
         distance ← CalculateDistance(CurrentLocationFromMobileUser(mu), GetLocation-
         FromIncidentEntry(incident))
         timestampDifference ← Timestamp.Now-GetTimestamp(incident)
         if distance ≤ NotifyDistanceThreshold && timestampDifference ≤ NotifyTimeThresh-
         old then
            SendPushNotification(mu)
         end if
      end for
   end for
end while
```

Figure 3.13. Notifier algorithm to send notification to mobile users.

4. PROTOTYPE IMPLEMENTATION

Prototype implementation of our proposed model, named EmNotifier consists of 3 parts. Front-end mobile application that users get emergency notification, front-end web application to analyse tweets in real time and to review incidents, back-end application for incident detection and notification generation. NoSQL (non-relational) database cluster have been used as data persistance layer. Queries that define an incident are given to system. At this step one thread establishes a single long live connection with Twitter. When an incident has been detected by back-end system, it generates a push message, determines the list of users to send push notification and forwards the message and list of users to Apple server. Front-end mobile application triggers, once user starts to move. At every 1km user change position, current GPS location of user send to EmNotifier server. Whenever, user moves into a region where an incident is happening, a notification will be triggered by mobile application. In implementation UpdateIncident method defined in our model takes the average of every related FeatureEnrichedPost location for incident center estimation and SemanticSearch method returns a popular location given a location entity. For instance, "Paris, France" will be returned by semantic sources for search query "Paris". This is due complexity of comparison for all possible location entities (see Section 3.5).

Front-end web application can be used to analyse tweets in near real time while collecting tweets, list all the incidents detected by EmNotifier, search collected tweets based on timestamp and incident type, annotating tweets for classification.

4.1. Front-end

EmNotifier consists of two different front-end applications; web front-end and mobile front-end. Section 4.1.1 and Section 4.1.2 gives detail on web and mobile front-end respectively.

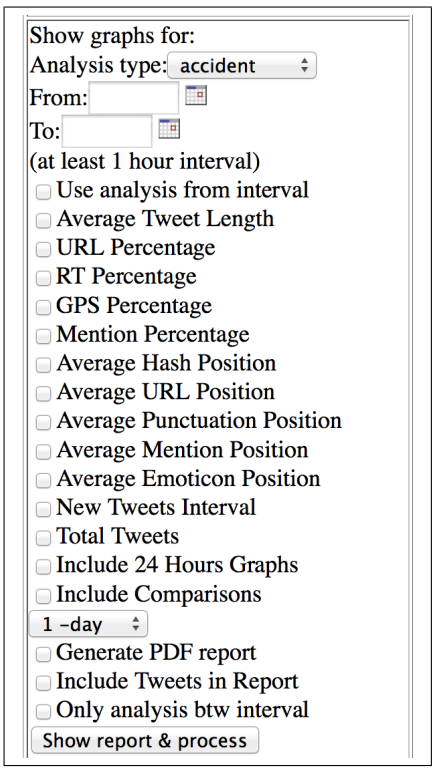


Figure 4.1. Prepare Customized Report in front-end web interface.

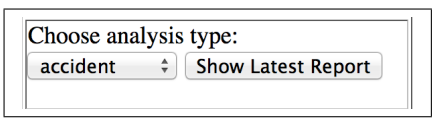


Figure 4.2. Show Latest Report in front-end web interface.

4.1.1. Web

Web front-end is developed in PHP language. It is used to analyse tweets, list all the incidents system has detected and to generate near real time tweet analysis reports.

4.1.1.1. Reporting. EmNotifier generates a report about sharing characteristics of collected tweets in 5 minutes interval while back-end fetches tweets. Some of the information depicted in the report are average tweet length, frequently used words, popular hashtags. All of the information can also be represented graphically. Figure 4.1 and Figure 4.2 shows the web interface for generating customized reports.

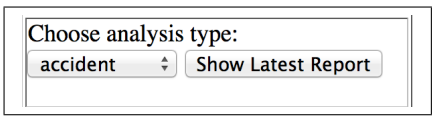


Figure 4.3. Search Tweets in front-end web interface.

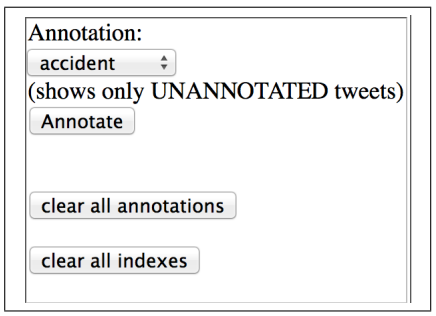


Figure 4.4. Annotation process in front-end web interface.

4.1.1.2. Search. While back-end fetches tweets, search can be carried out on collected tweets using web front-end based on incident type and timestamp. Figure 4.3 shows the search for tweets menu.

4.1.1.3. Annotation. In order to identify Twitter contribution characteristics, we have manually annotated tweets. Besides that, training dataset is necessary for supervised classifier and NER. Figure 4.4 shows the web interface to initialize annotation process for any type of incident. Tweets are retrieved from database according to FIFO principle for annotation.

In some cases, especially after learning the exact time of incident, it is more likely to examine tweets based on only a certain time range to increase the possibility of finding a set of incident related tweets (see Figure 4.5). Tweets can be annotated according to their timestamp (see Figure 4.6).

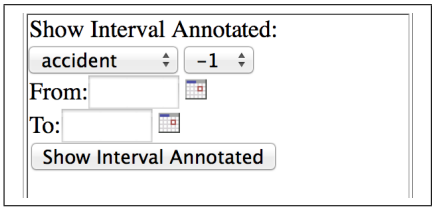


Figure 4.5. Show interval annotated in front-end web interface.

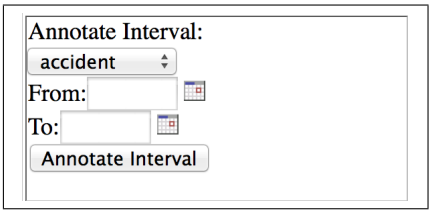


Figure 4.6. Annotate interval in front-end web interface.

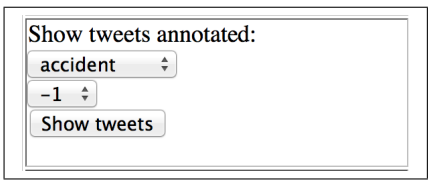


Figure 4.7. Show tweets annotated in front-end web interface.

Tweets that are annotated can be examined according to class (positive/negative) and incident type that they belong to (see Figure 4.7).

4.1.1.4. Training/Test sets. After annotation process, preparation of training/test set is necessary for supervised classification. Tweets can easily be assigned to training/test set by applying a filter on classes that they belong to (positive/negative). Tweets that belong to a test/training set can be examined again (see Figure 4.8).

4.1.1.5. View Incidents. Incidents detected by EmNotifier are stored in database. "View Incidents" menu (see Figure 4.9) can be used to list incidents based on incident type.

4.1.2. Mobile

Mobile front-end is written in Objective-C and runs on Apple IOS[15] . Mobile application allows users to receive emergency notification. Once EmNotifier detects the incident and estimate its respective location, it sends push notification to users within 1km range of incident. Mobile application can run both in background and foreground. In case of background operation, on-screen information about the incident

[15] https://developer.apple.com/devcenter/ios/index.action

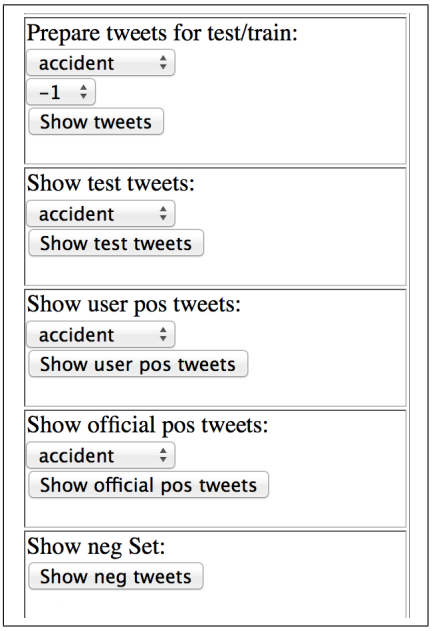


Figure 4.8. Training/Test set preparation in front-end web interface.



Figure 4.9. View incidents in front-end web interface.



Figure 4.10. Mobile Notification.

is shown in a notification box (see Figure 4.10). If the application was operating on foreground once user got notification, the estimated location of incident is shown on map in addition to additional information (see Figure 4.12).

As we have discussed in our model, the purpose of our system is to notify user about an ongoing incident where;

- User is nearby to incident location.
- User is interested in incident location.

User can add pre-defined cities and countries as interest location to his/her profile (see Figure 4.13). A notification will be send to user once an incident breaks out in one of the locations user interested in.

User can easily understand the type of incident happening via representative graphics. Our analysis show that, in some of emergency situations microbloggers post incident related images. For instance, as we have discussed in Section 6, users attached fire incident related images to their tweets while the incident was going on. Rather than a simple graphic representation of incident, live images about the incident can be shown to user. User can share information about the latest incident to his/her acquaintances via e-mail, using the "Share" button.

4.1.3. Main technologies for Front-end

The two front-end uses different technologies and notable ones are listed below.

- Web front-end is written in PHP language. For graphical representation of several values, pChart[16] library have been used.
- Web front-end uses MongoDB as data persistance layer. In order to communicate with MongoDB, mongodb-php library[17] is necessary.

[16] `http://pchart.sourceforge.net/`
[17] `http://docs.mongodb.org/ecosystem/drivers/php-libraries/`

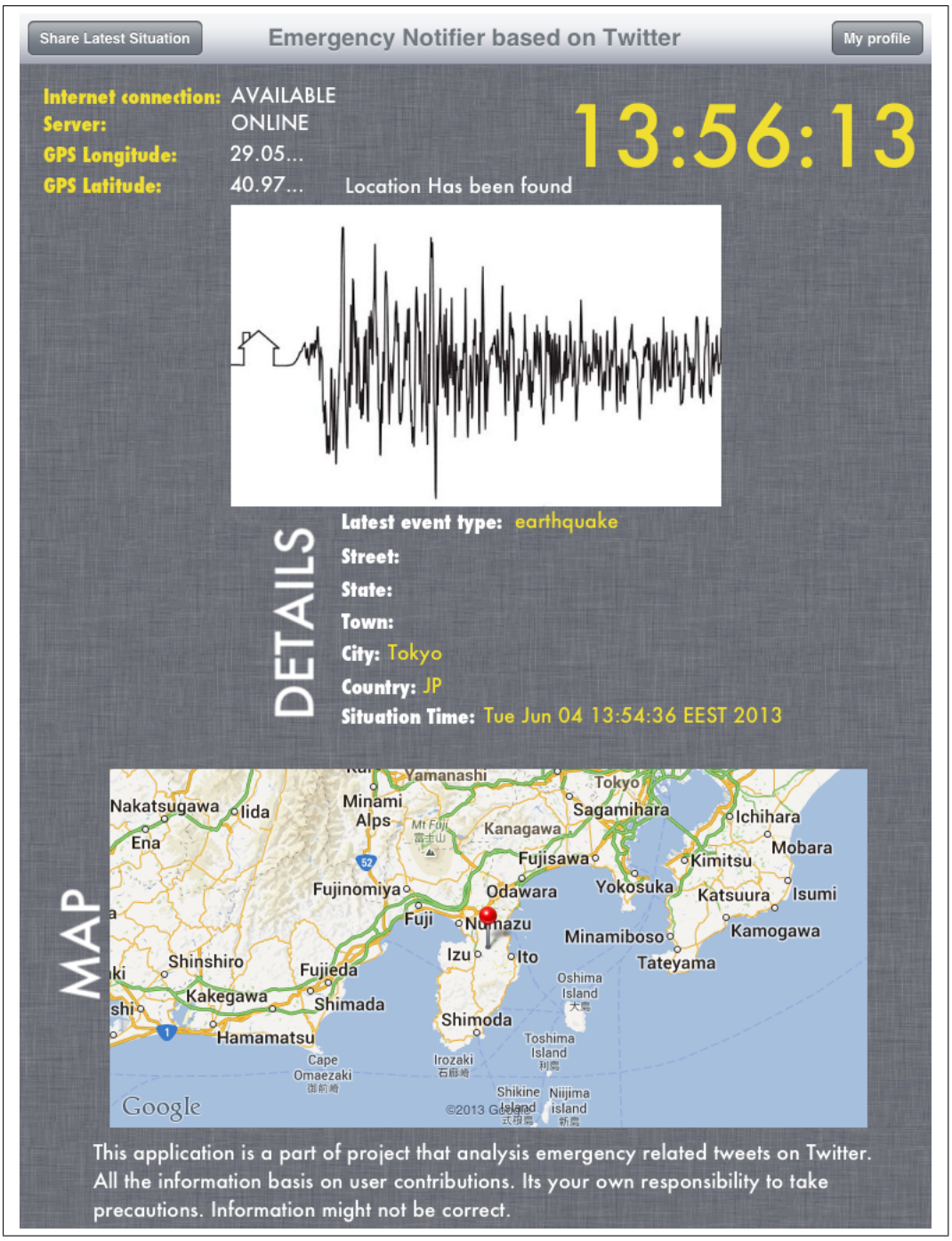


Figure 4.11. Mobile Front-end in case of earthquake incident.



Figure 4.12. Mobile Front-end in case of fire incident.



Figure 4.13. Mobile Application Customization.

- Mobile front-end is written in Objective-C language. A valid Apple developer account[18] is necessary to test application on a physical device. Mobile application is tested on Apple iPad 2 device running IOS 5.1. As an internal database, it uses SQLite.[19]
- Apple Push Notification Service[20] is used both on mobile front-end and back-end for receiving and pushing notification. Notification certificate[21] for mobile application should be generated once in order to push/receive notifications.

4.2. Back-end

EmNotifier back-end is written using Java programming language. It uses a document-oriented NoSQL database named MongoDB.[22] Section 4.2.1 mentions about database design and Section 4.2.2 focuses on application layer of EmNotifier system.

4.2.1. Database

Today, MongoDB gets high attention and becomes popular like the traditional relational database MySQL. All the tweets collected from Twitter system are stored in MongoDB. Rather than just tweets, the tweet analysis, incident and user related information is stored on MongoDB as well. MongoDB is capable of handling JSON/BSON type of tuples. Users might directly insert a JSON document and index multiple fields of it. In document-oriented non-relational database system, the term collection refers to table in relational database and the term document refers to tuple.

4.2.1.1. Design. EmNotifier database consists of 3 config, 2 replica, 1 arbiter servers. Although it is expected to run the servers separately as 6 machines, we have 3 machines that undertake multiple functionalities. Figure 4.14 shows our database system consisting of 3 machines.

[18] https://developer.apple.com/programs/
[19] http://www.sqlite.org/
[20] http://tinyurl.com/lku26e3
[21] http://tinyurl.com/lbahdg8
[22] http://www.mongodb.org/

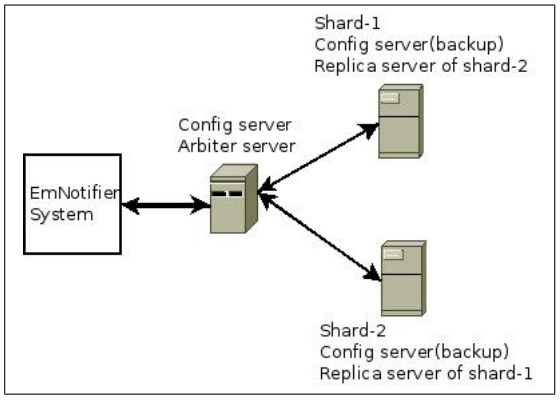


Figure 4.14. EmNotifier database design.

Sharding refers to split the data into parts and store them separately for query performance and storage space necessities. Config server redirects database query request from EmNotifier to one of the shards. Data is stored in shards according to their object IDs. Query is directed to one of the shards based on object IDs. For example, object IDs between 1-10 stored in shard 1, object IDs between 11-20 in shard 2. This is automatically configured by MongoDB.

Config server takes the responsibility to direct incoming requests from EmNotifier between servers. In case of failure, one of the two other config servers take the responsibility for traffic management. Replica server stores the backup of shard server data. If shard server fail, replica of the shard server automatically becomes primary. As an example from our design, if machine-1 fails, its replica is stored on machine-2. This way queries that would be directed to shard-1 on machine-1 will be directed to replica of it, that resides on machine 2. Whenever, the machine-1 goes online, shard-1 on machine-1 will be primary, whereas replica of it on machine-2 will be secondary. In order to chose which machine to be primary or secondary in a multi server system, a voting mechanism is necessary. A successful voting would be between odd number of servers. Arbiter server, plays the role for just voting, to chose the primary and secondary servers.

4.2.1.2. Schema. EmNotifier persists its data on MongoDB. In this section collections and fields of our database schema is given.

collection_analysis: EmNotifier perform tweet analysis for every 5 minutes once the system fetches tweets in real-time. CollectionAnalysis store analysis related data on MongoDB. Fields of the collection are;

- *incidentType:* Incident type that analysis performed on.
- *averageTweetLength:* Average tweet length of stored tweets.
- *averageTweetLengthFiltered:* Average tweet length after stop word elimination.
- *mostActiveUsers:* Most contributed users.
- *topNHour:* Highest number of tweets collected per hour.
- *topNWords:* Frequently used N words.
- *topNEmoticons:* Frequently used N emoticons.
- *topNHashtag:* Frequently used n hashtag.
- *topNPunctuation:* Frequently used n punctuation.
- *percentageURL:* Percentage of URL in post.
- *percentageRT:* Percentage of retweets.
- *percentageGPS:* Percentage of GPS attached to tweets.
- *percentageMention:* Percentage of mention.
- *totalTweets:* Total number of tweets collected.
- *averageHashLocation:* Average position of hashtag(s) in tweet text.
- *averageURLLocation:* Average position of URL(s) in tweet text.
- *averagePuncLocation:* Average position of punctuation(s) in tweet text.
- *averageMentionLoc:* Average position of mention(s) in tweet text.
- *averageEmoticonLoc:* Average position of emoticon(s) in tweet text.
- *newTweetsinterval:* Number of tweets collected within 5 minute interval.
- *percentageInCity:* Percentage of location as type of city found in tweet text.
- *percentageInCountry:* Percentage of location as type of country found in tweet text.
- *percentageInTown/State:* Percentage of location as type of town found in tweet text.
- *percentageInAddress:* Percentage of location address found in tweet text.
- *percentageInAbb:* Percentage of location as type of abbrevation found in tweet

text.

- *percentageInGPS:* Percentage of location as type of GPS found in tweet text.
- *percentageOutSarcastic:* Percentage of sarcastic location found in user profile.
- *percentageOutCountry:* Percentage of location as type of country found in user profile.
- *percentageOutTown* Percentage of location as type of town found in user profile.
- *percentageOutCity:* Percentage of location as type of city found in user profile.
- *percentageOutAddress:* Percentage of location as type of address found in user profile.
- *percentageOutAbb:* Percentage of location as type of abbrevation found in user profile.
- *percentageOutGps:* Percentage of location as type of GPS found in user profile.

collection_fep: CollectionFep stores data for feature enriched posts. Fields of the collection are;

- *annotated:* Flag indicating if FeatureEnrichedPost is annotated manually.
- *classifiedAs:* Classifier result for FeatureEnrichedPost. Values are "positive", "negative".
- *incidentType:* Incident type of FeatureEnrichedPost that is determined manually.
- *hasSet:* Flag indicating if FeatureEnrichedPost belongs to a test/train set.
- *inSet:* Set(test/train) that the FeatureEnrichedPost belongs.
- *status:* Manually determine FeatureEnrichedPost class. Values are "positive", "negative".
- *raw_tweet*
  (i) *timestamp:* Timestamp of tweet.
  (ii) *follower:* Follower count of user.
  (iii) *following:* Following count of user.
  (iv) *location profile:* Profile location manually entered by user.
  (v) *originalPost:* Tweet text.
  (vi) *filteredPost:* Stop word eliminated tweet text.

(vii) *hashtagArray:* Hashtags in tweet text.

(viii) *emoticonArray:* Emoticons in tweet text.

(ix) *mentionArray:* mentions in tweet text.

(x) *punctuationArray:* punctuations in tweet text.

(xi) *timezone:* Timezone written in users profile.

(xii) *tweetId:* Tweet id given by Twitter.

(xiii) *screenName:* Screen name of user.

(xiv) *tweetLength:* Original tweet length.

(xv) *tweetLengthFiltered:* Stop word eliminated tweet length.

(xvi) *geoLocation:* Geo location of user determined by attached GPS to tweet.

collection_mobileuser: Stores data for users that installed mobile application of EmNotifier. Fields of the collection are;

- *DeviceToken:* Embedded IOS device token.
- *currentLatLong:* Current latitude/longitude of user.
- *interestCount:* Total number of interested locations of user.
- *interestLatLongArray:* GPS positions of users interested locations.
- *latestIncidentNotifyCity:* Latest incidents city information for a user interested location.
- *latestIncidentType:* Latest incident type for a user interested location.
- *latestIncidentLatlong:* Latest incidents GPS information for a user interested location.
- *latestIncidentNotifiyCountry:* Latest incidents country information for a user interested location.
- *latestIncidentNotifiyStreet:* Latest incidents street information for a user interested location.
- *latestIncidentNotifiyTimestamp:* Latest incidents timestamp information for a user interested location.

collection_incidents: Stores all incidents detected by EmNotifier. Fields of the

collection are;

- *tweetIdArray:* Tweets that are related to incident.
- *incidentId:* Incident ID.
- *country:* Incident country information.
- *city:* Incident city information.
- *state:* Incident state information.
- *town:* Incident town information.
- *street:* Incident street information.
- *latlong:* Incident GPS information.
- *incidentType:* Incident type information.

collection_timezones: Timezone and city information collected from IANA timezone database.[23] Fields of the collection are;

- *countrycode:* Country code. For example: TR, GB, DE.
- *continent:* Continent the country belongs. For example: Africa, Asia, Europe.
- *city:* Cities that belong to timezone.
- *gmtoffset:* Absolute time reference. For example: +02:00,+03:00.

collection_wordmap: For every unique word, CollectionWordmap stores words and related IDs. Fields of the collection are;

- *wordid:* Unique word id.
- *value:* Word itself.

4.2.2. Application Layer

We have used several libraries in EmNotifier system. In order to run EmNotifier successfully, all the libraries depicted below should be installed. Back-end of

[23]IANA database: `http://www.iana.org/time-zones`

EmNotifier system runs on Ubuntu Linux 13.04.[24] At least 1GB of RAM is necessary to execute the application. Classes written for EmNotifier given in Table 4.1.

- Twitter4J[25] is used to communicate with Twitter microblogging system. This library is the best and easiest way to fetch tweets from Twitter. Search API or StreamingAPI connection can easily be set up. Documentation is well written and it is updated frequently as Twitter makes structural changes to data.
- Apache Lucene 3.6.2[26] is used for indexing tweets. We have generated text analysis related reports near real time by indexing several fields of FeatureEnrichedPost.
- Apache Mahout[27] have been used for determining collocation from tweets. Due incompatability issues we have encountered with Java, binary terminal application is called within Java.
- DBPedia Spotlight [28] is used to communicate with DBPedia. Web Service of DBPedia is installed on a single machine in order not to get blacklisted by DBPedia servers with frequent requests.
- MapQuest OpenStreetMap API[28] have been used for the purposes of reverse-geocoding and geocoding. Although the company does not specify a usage limitation like the OpenStreetMap does (1 requests per minute), we have designed the back-end not to send frequent requests. In addition to that, batch geocoding function of the API is used.
- CMU Twitter NLP[29] is a library of part-of-speech tagging for Twitter. It is used to tag tokens as emoticon, punctuation, hashtag, URL, mention.
- JavaPNS[30] is the library used to send notification through Apple Push Notification Service to IOS devices.

Limitation 4.1. *We are using a 3rd party application, named MapQuest for geocoding and reverse geocoding. Unfortunately, this situation forces us to limit the frequency*

[24] http://www.ubuntu.com
[25] http://twitter4j.org/en/index.html
[26] http://lucene.apache.org/core/
[27] http://mahout.apache.org/
[28] http://developer.mapquest.com/
[29] http://www.ark.cs.cmu.edu/TweetNLP/
[30] http://code.google.com/p/javapns/

Table 4.1. Java classes developed in EmNotifier.

Class	**Description**
Classification	Prepare feature vector, classify given text
Coordinate	Represens a coordinate in decimal degrees
DBConnect	Database connection, insert, update functions
HighFreqTerms	Extracts the top n most frequent terms from an existing Lucene index
IndexAnalyzer	Real time text analysis
Indexer	Indexes given data into index directory
LocationManager	Location extraction related functions
Notifier	Push message to interested users about an ongoing incident
StreamReader	Establishes connection to Twitter, fetching Query related tweets
TextAnalytics	POS tagging, StopWordElimination, NER

of geocoding. However, this problem can be solved by installing a complete dump of DBpedia:Place, or LinkedGeoData on personal server.

Limitation 4.2. *Apple Push Notification Service have been used to send notification to users. Push notifications are unreliable because the principle for sending notification is fire and forget. After pushing notification from back-end to APNS, delivery of the notification depends on the APNS server. However, in our tests (see Table 5.15) the result was under 10 seconds. If the users mobile is not connected to internet, or if required ports are blocked for APNS connection (ex: public WiFi hotspot) the notification could not be send. Once user gets online APNS pushes all the notifications that could not be send to user again for limited time.*[31]

[31] `http://tinyurl.com/44f7kop`

5. EVALUATION

In this chapter, we are evaluating our model using real data collected from Twitter system. Section 5.1 discusses named entity recognizer performance for location entity detection. Classification performance for earthquake and fire incident related tweets have been evaluated in Section 5.2. Real time incident detection performance for fire and earthquake incidents have been discussed in Section 5.3. In Section 5.4, we have experimented processing duration of a single tweet for location extraction and notification delay on 3 mediums; Public Wifi, Edge and 3G.

5.1. NER Location performance

We have manually annotated 1550 microblog posts to train named entity recognizer for location named entity recognition. Out of this 1550 annotations, 933 were related to earthquake and 617 fire incidents. Every microblog post in the dataset contains one location entity. We have tested the NER performance using 10-fold cross validation. In every iteration random 1395 microblog posts were used for training and 155 for testing. F-score of trained named recognizer is 0.811. Recall and precision are 0.765 and 0.862 respectively.

Table 5.1. Sample NER annotations.

Sample NER annotation
Earthquake in <START:LOCATION> Tirana <END>
Fire burning near <START:LOCATION> Lake Minnequa <END> http://t.co/qHNW8q6j
Earthquake in <START:LOCATION> Taiwan <END> !
The gas station that's in fire, on <START:LOCATION> Amboy Ave <END> looks wild !
Firefighters Respond To Large House Fire In <START:LOCATION> Oklahoma <END> http://t.co/UWvvkMXQ

5.2. Classification performance

We have evaluated incident classification performance of our proposed feature vector using Naive Bayes and SVM methods with 2-fold, 5-fold, 10-fold cross validation. We have manually annotated 1300 earthquake and 1020 fire incident related microblog posts. We have trained, tested the performance of classification separately for fire and earthquake incidents. We have made a comparison whether there is an improvement for classification using our proposed feature vector versus feature vectors proposed by other studies.

We have used the same training and test set while evaluating both Naive Bayes and SVM classifier. As it can be seen from Table 5.2 and 5.3, SVM classifier scored higher than Naive Bayes classifier based on average f-score using 2,5,10 fold cross validation.

Our proposed feature vector (feature set-1) consists of; stop word count, tag frequency, position and boundary tokens for query words, location entity, emoticon, hashtag, url, punctuation in post (see Section 3.3 for detailed information). Second

feature vector (feature set-2) consists of token count and position of query words in post.

Including specific features of microblogging system contributions in feature vector as; hashtag, URL, mentions, emoticons, punctuations improved the classification performance compared to a feature vector that does not take microblogging system specific features (see Table 5.4, 5.5). We have discussed the use of microblogging system specific features for fire and earthquake related incidents detailed in Section 6.

Table 5.2. Average F-score comparison of earthquake incident for Naive Bayes and SVM classifer.

N cross validation	**Avg. F-Score SVM**	**Avg. F-Score Naive Bayes**
2	0.877	0.856
5	0.892	0.860
10	0.899	0.863

Table 5.3. Average F-score comparison of fire incident for Naive Bayes and SVM classifer.

N cross validation	**Avg. F-Score SVM**	**Avg. F-Score Naive Bayes**
2	0.882	0.866
5	0.865	0.831
10	0.862	0.802

Table 5.4. Average F-score comparison of earthquake incident for Naive Bayes and SVM classifer with feature set-1 and feature set-2.

	Average F-score			
	SVM		**Naive Bayes**	
N cross validation	**Feature set-1**	**Feature set-2**	**Feature set-1**	**Feature set-2**
2	0.906	0.877	0.858	0.837
5	0.892	0.824	0.860	0.825
10	0.910	0.899	0.863	0.808

Table 5.5. Average F-score comparison of fire incident for Naive Bayes and SVM classifer with feature set-1 and feature set-2.

	Average F-score			
	SVM		**Naive Bayes**	
N cross validation	**Feature set-1**	**Feature set-2**	**Feature set-1**	**Feature set-2**
2	0.882	0.834	0.866	0.852
5	0.895	0.845	0.842	0.831
10	0.861	0.821	0.816	0.802

5.3. Incidents detected

In this section, we examine the incident detection performance of our system. In order to detect fire and earthquake incidents our system run at different times for a certain period. We have run the system limited periods due to use of 3^{rd} party systems (see Section 4.2.2) and the amount of free space on disk. In Table 5.6 and Table 5.9 we have depicted incident occurrences detected by EmNotifier. For both of the tables ΔLocation is the difference between the actual location of incident, computed location and ΔT is the difference between the actual time of incident, the time EmNotifier take notification decision.

The system continuously run to detect fire incidents at times between 24/05/2013 22:47 - 25/05/2013 01:33, 25/05/2013 09:36 - 25/05/2013 23:38, 26/05/2013 20:00 - 27/05/2013 00:43, 27/05/2013 14:00 - 28/05/2013 01:28. Total of 337,128 tweets have been fetched from Twitter system. Additionally, system continuously run to detect earthquake incidents at times between 24/05/2013 10:00 - 24/05/2013 22:45, 26/05/2013 09:32 - 26/05/2013 16:02, 28/05/2013 11:00 - 28/05/2013 22:40, 30/05/2013 21:00 - 30/05/2013 22:00, 31/05/2013 23:00 - 01/06/2013 19:50. Total of 273,843 tweets have been fetched from Twitter system.

Our system run real time for incident detection using SVM classifier, trained by 1300 earthquake, 1020 fire and 2000 incident unrelated tweets. Threshold for SVM classifier is 0.90.

Definition 5.1. Incident candidate *EmNotifier did not take notification decision.*

Definition 5.2. Incident occurrence *Notification decision taken by EmNotifier.*

While the system was running in real time between the given time periods, system identified 1,146 earthquake incident candidates. 25 of the incident candidates are determined as incident occurrences. We have set the notification threshold for earthquake incident as 3, based on the average count of contributions per incident. We have determined notification time threshold for earthquake incident as 2 minutes, depending

on duration to reach average count of incident related tweets. Australian earthquake center[32] defines a term named felt radius. We have examined several earthquake felt radiuses and determined distance threshold for earthquake incident as 100Km.

We have determined the correctness of earthquake incident occurrences by an earthquake report[33] web site. Specified web site aggregates and broadcasts earthquake occurrences from USGS[34] , GEOFON[35] , EMSC[36] and PTWC[37] . We think that some of the incident candidates can be verified by local seismic centers. For instance we were able to verify Pakistan earthquake from Pakistan Seismic center, whereas it was not mentioned by USGS.

As we have denoted in Table 5.6 our closest estimation of earthquake epicenter verified by official sources is 2.90km, farthest 2225.22km and on average 393.05km. When we filter incorrect location estimations based on country and compare with trustworthy sources, our closest estimation of earthquake epicenter verified by official sources is 2.90km, farthest 708.11km and on average 218.79km.

Sensitivity of location information we were able to extract from a contribution is of great importance for location estimation. Discussed in Section 6 high sensitive location information rate is low, whereas low sensitive location information is high. In addition to location sensitivity, distance of earthquake epicenter to places of residence is another factor to estimate the epicenter of earthquake by human sensors.

For earthquake incident, system generates a notification maximum 9 minutes, minimum 2 minutes and on average 6 minutes after the first post. First post about the earthquake incident by human sensors were reported at minimum 30 seconds, maximum 5 minutes and on average 1 minutes after the incident. First response time for an earthquake incident changes according to the severity of the earthquake felt by human

[32] http://www.ga.gov.au/
[33] http://earthquake-report.com/
[34] U.S. Geological Survey, http://www.usgs.gov/
[35] Global Seismic Network, http://geofon.gfz-potsdam.de/
[36] European-Mediterranean Seismological Centre,http://www.emsc-csem.org/
[37] Pacific Tsunami Warning Center, http://ptwc.weather.gov/

sensors. We can say that even for small scale incidents contributors post tweets.

We have determined that a total of 246 greater than 2.5 richter scale earthquake incidents occurred while the system was running real time. 203 of 246 earthquake occurrences are 2.5-4.5 richter scale. 43 of 246 earthquake occurrences are 4.5 and greater richter scale. Our system detected 10, 4.5 and greater richter scale and 15, 2.5-4.5 richter scale earthquake occurrences. We have observed that contributors reported even for small richter scale earthquake occurrences.

For fire incident EmNotifier identified 857 incident candidates. Once we have examined manually 857 incident candidates, 116 of them were describing a fire incident. 85 of 116 incident candidates were verified by trustworthy sources. Although we have used trustworthy sources for verification, we were not able to find the actual incident time for several incidents. Out of 85 verified incident information, 58 of them were containing actual time of incident. For that reason, in Table 5.6 we have just written incident occurrences that are verified by trustworthy sources and actual time of incident is known.

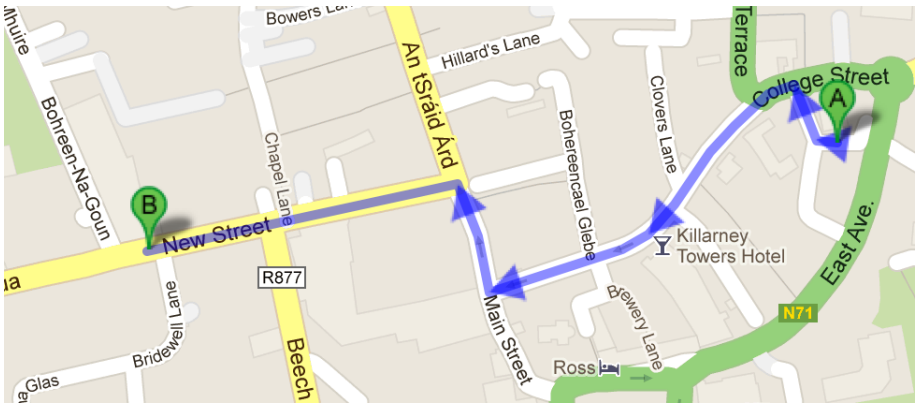
Out of 116 incident candidates EmNotifier take notification decision for 42 incident occurrences and generated notification for users 1km around the location of incident occurrence. Based upon these 42 incident occurrences, a notification have been sent to interested users at minimum 3 minutes, maximum 36 minutes and on average 15 minutes after the first post about the incident.

Posting period of notification depends the number of contributions for incident. We have set the Notification threshold as 2 for fire incident, by examining and taking the average of contributions per incident in our dataset. Comparing contribution frequency per minute for earthquake incident to fire incident, contributions for local fire incidents is low. In case of earthquake incident, contributors repeatedly post immediately after the incident whereas in case of fire incident, contributors should have to notice the presence of a fire incident for a contribution.

According to descriptions of official sources we have determined the actual location of incident. For all incident occurrences that are verified by trustworthy sources when we compare actual location to our location estimation; at minimum 2 meters, farthest 698km and on average 38.19km distant from the actual location of incident. Note that in Table 5.6 we have only evaluated incident occurrences that are verified sources that contain location information and actual time of incident.

Location estimation accuracy compared to the actual location of incident increases as the location sensitivity per contribution increases. For instance, an incident incurred on M1 highway, South Africa have been location estimated by system as one of the popular point in South Africa. South Africa term have been extracted from in post. For this reason, distance from the center of the incident are exaggerated. Most closely identified fire incident that is 2 meters close to the actual location of incident have been detected by the help of GPS information attached to contribution by user. We have verified the presence of a fire incident by online newspapers and several fire department web sites. Reason we have use online newspapers is due some of the local fire departments does not share fire incident information on their web sites. However, we have found several fire departments that share local fire incident related information. These are London[38] , Manchester[39] , South Yorkshire[40] , Bucks[41] and Lancashire[42] fire departments web sites. We estimate there were 128 fire incidents reported by given fire departments at the times our system were running. Our system was able to detect 6 of them. First posts about the fire incident are shared at minimum 1 minutes, longest 6 hours and on average 1:38 hours after the incident. Some fire incidents are failed to get under control after long time. Contributors share updated information about the incident such as “fire is still going on...!”. Additionally, contributors might share “black smoke rising from the park!” while they are passing nearby the incident region afterwards.

[38] http://www.london-fire.gov.uk/
[39] http://www.manchesterfire.gov.uk/updates/incidents.aspx?page=0
[40] http://www.syfire.gov.uk/LatestIncidents.asp
[41] http://tinyurl.com/mh9uu6e
[42] http://www.lancsfirerescue.org.uk/latest-incidents/



Microblog post: Major fire on Killarney's New Street. Traffic at a standstill so avoid, if possible

Timestamp for first post: 25/5/13 15:46:46

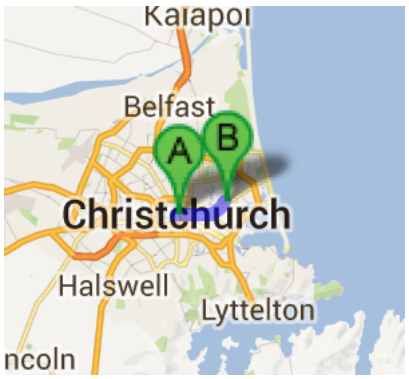
Actual time of incident: 25/5/13 15:45

Δ*Location:* 700m

Point A: Location computed by EmNotifier

Point B: Actual location of incident

Figure 5.1. Location detected by EmNotifier for fire incident.



Microblog post: Earthquake

Timestamp for first post: 28/5/13 21:54:53

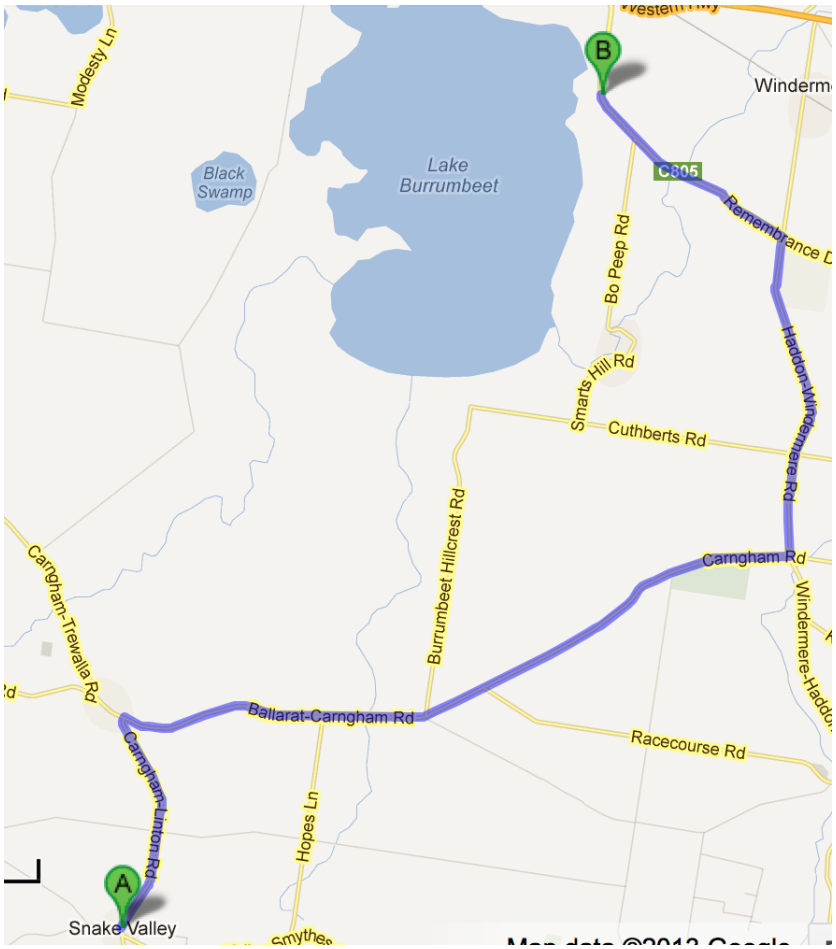
Actual time of incident: 28/5/13 21:54:17

Δ*Location:* 2.896km

Point A: Earthquake epicenter

Point B: Location computed by EmNotifier.

Figure 5.2. Location detected by EmNotifier for earthquake incident.



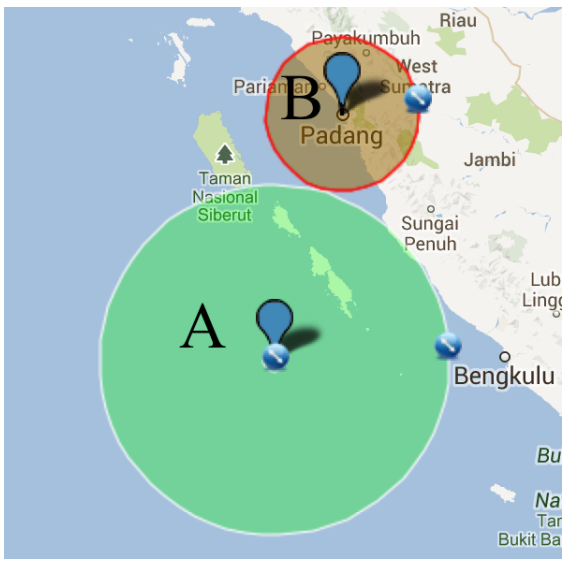
Raw contribution: #chepstowe fire heading straight towards burrumbeet - cfa hoping to stop it before it reaches lake

Point A: Ballarat near Snake Valley

Point B: Burrumbeet Caravan Park[a]

[a] http://news.cfa.vic.gov.au/news-in-brief/item/436-kentbruck-bushfire-update.html

Figure 5.3. Importance of notification fire.



Earthquake information: 30/5/13 21:37, Southwest Of Sumatra, Indonesia(-3.9358,99.529), magnitude: 5.3

Computed location: Padang, West Sumatera (INA)(-0.95, 100.3530556)

Δ*Location:* 344.394Km

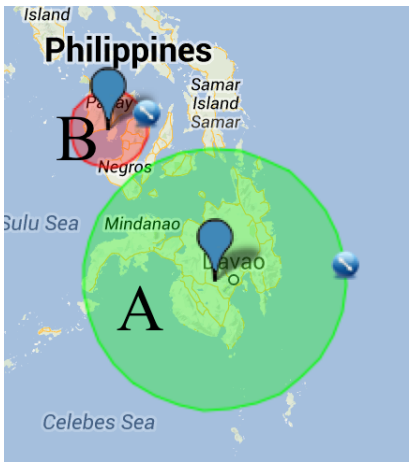
Felt radius[a] *:* 228Km

Region-A: Actual earthquake epicenter and felt radius

Region-B: Notified area, 100Km around computed location

[a] http://www.ga.gov.au/earthquakes/getQuakeDetails.do?quakeId=3370081&orid=768236&sta=N/A

Figure 5.4. Notified area sample-1 in case of earthquake.



Earthquake information: 01/06/2013 14:10:11, Mindanao, Philippines(7.215, 124.839) magnitude: 5.6

Computed location: Paradise Cagayan De Oro, Philippines(10.69,122.54)

Δ*Location:* 461.577km

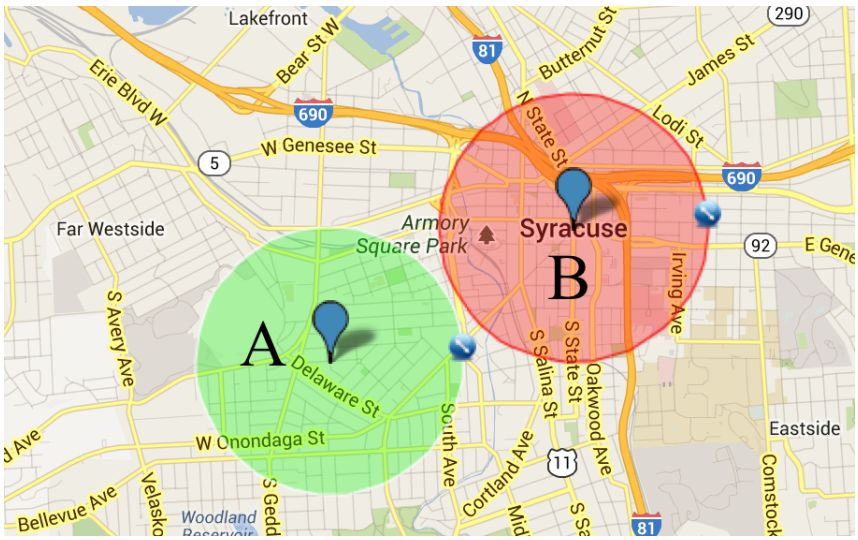
Felt radius[a] *:* 340km

Region-A: Actual earthquake epicenter and felt radius

Region-B: Notified area, 100km around computed location

[a] hhttp://www.ga.gov.au/earthquakes/getQuakeShakeDamage.do?quakeId=3371381

Figure 5.5. Notified area sample-2 in case of earthquake.



Fire information: 24/5/13 22:05, Davis Street, Syracuse, NY 13204, USA(43.039217, - 76.169803)

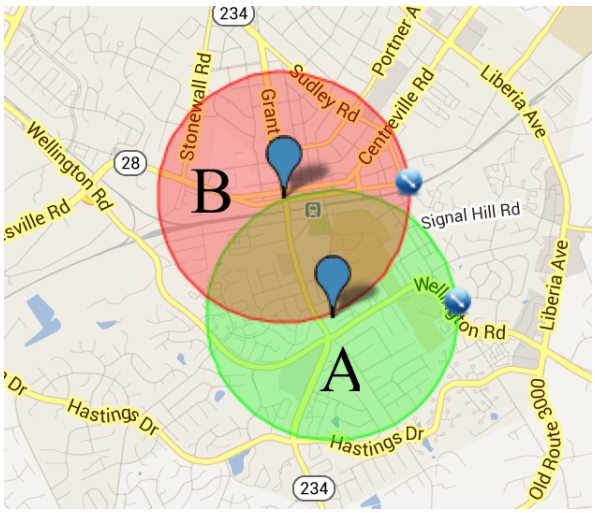
Computed location: Syracuse, NY, USA(43.0481221, -76.1474244)

Δ*Location:* 2.071km

Region-A: 1km around actual fire location

Region-B: Notified area, 1km around computed location

Figure 5.6. Notified area sample-1 in case of fire.



Fire information: 25/5/13 13:45, Sandalwood Drive, Manassas, VA 20110, USA(38.7424903, - 77.4708337)

Computed location: Manassas, VA, USA(38.7509488, -77.4752667)

Δ*Location:* 1.016km

Region-A: 1km around actual fire location

Region-B: Notified area, 1km around computed location

Figure 5.7. Notified area sample-2 in case of fire.

Table 5.6 denotes earthquake occurrences detected by EmNotifier and verified by trustworthy sources (Min. decision time=1min, Max. decision time=13min, Avg. decision time=7min. Min. location estimation=2.90km, Max. location estimation=2225.22 km, Avg. location estimation=393.05km).

Table 5.6. Earthquake incident occurrences detected by EmNotifier and verified by trustworthy sources (24/5/13 10:20 - 28/5/13 21:54).

Incident time	**Richter scale**	**Actual Location**	**Computed Location**	**ΔLocation (km)**	**Computed Incident time**	**ΔT (min)**
24/5/13 10:20	2.7	Northern California (40.1862,-121.069)	Carmichael, CA, USA (38.6171, -121.3282)	175.889	24/5/13 10:29	9
24/5/13 10:38	2.6	Northern California (40.1768,-121.046)	Oakland, CA, USA (37.8043, -122.2711)	284.250	24/5/13 10:48	10
24/5/13 11:02	4.9	Northern California (40.1717,-121.082)	Northern California (38.8375, -120.8958)	149.211	24/5/13 11:06	4
24/5/13 11:15	2.7	Northern California (40.184,-121.074)	North Hollywood, Los Angeles, CA, USA (34.1870,-118.3812)	708.111	24/5/13 11:28	13
24/5/13 11:33	5.2	Near East Coast Of Honshu, Japan (37.77,141.6)	Tokyo, Japan (35.6894, 139.6917)	287.110	24/5/13 11:38	5
24/5/13 11:58	4.5	Minahasa, Sulawesi, Indonesia (0.73,122.76)	Modelomo, 96263, Indonesia (0.5191, 122.3376)	52.482	24/5/13 12:06	8
24/5/13 18:28	3.9	Northern California (40.1785,-121.056)	Orinda Woods Drive, Orinda, CA 94563, USA (37.8842, -122.1839)	273.047	24/5/13 18:36	8
24/5/13 21:22	2.9	Northern California (40.1872,-121.062)	Las Vegas, NV, USA (36.1146, -115.1728)	685.457	24/5/13 21:28	6
28/5/13 11:16	3.6	Hastings (-39.7824,177.107)	Hawkes Bay, New Zealand (-39.7711, 176.7416)	31.248	28/5/13 11:22	6
28/5/13 21:54	3.4	Christchurch (-43.5386,172.671)	Christchurch, New Zealand (-43.5320, 172.6362)	2.896	28/5/13 22:02	8

Table 5.7. Earthquake incident occurrences detected by EmNotifier and verified by trustworthy sources (30/5/13 21:37 - 1/6/13 14:49).

Incident time	**Richter scale**	**Actual Location**	**Computed Location**	**ΔLocation (km)**	**Computed Incident time**	**ΔT (min)**
30/5/13 21:37	5.3	Southwest Of Sumatra, Indonesia (-3.9358,99.529)	Padang, West Sumatera (INA) (-0.95, 100.3530)	344.394	30/5/13 21:46	9
31/5/13 00:04	2.6	105 km NW of Jhang, Pakistan (31.95, 71.5)	Lahore, Pakistan (31.5455, 74.3406)	272.347	31/5/13 00:12	8
31/5/13 00:52	2.5	Southern Alaska(60.1105,-152.736)	Vancouver, BC(49.2612,-123.1139)	2225.224	31/5/13 00:59	6
31/5/13 01:10	4	(14.6878, -61.0691)	Trinidad & Tobago(10.691, -61.2225)	444.650	31/5/13 01:17	7
31/5/13 05:40	2.4	2.4 1km NW of West Rancho Dominguez, California(33.908, 118.283)	Los Angeles, CA, USA(34.0522, 118.2436)	16.442	31/5/13 05:46	6
31/5/13 07:48	2.9	14km SW of Westwood, California (40.3059, -121.0057)	Kansas,USA(39.0119, -98.4842)	1928.095	31/5/13 07:56	8
31/5/13 09:05	4.6	Mona Passage, Dominican Republic(18.0561,-68.3554)	(18.8360,-69.5238)	150.703	31/5/13 09:14	9
31/5/13 20:36	3.9	30 km south-east of Amberley(-43.37, 172.99)	New Zealand(-40.9005, 174.8858)	315.954	31/5/13 20:46	10
1/6/13 11:43	2.7	Northern California(40.1753,-121.109)	Northern California(38.8375, -120.8958)	149.874	1/6/13 11:49	6
1/6/13 14:49	4.8	Tor,Egypt (28.390,33.327)	Cairo, Cairo Governorate, Egypt(30.0444, 31.2357)	273.910	1/6/13 14:53	4

Table 5.8. Earthquake incident occurrences detected by EmNotifier and verified by trustworthy sources (1/6/13 16:28 - 1/6/13 19:35).

Incident time	**Richter scale**	**Actual Location**	**Computed Location**	**ΔLocation (km)**	**Computed Incident time**	**ΔT (min)**
1/6/13 16:28	3.6	South Parganas,West Bengal(22.1,88.7)	Kolkata-India(22.5726, 88.3638)	62.905	1/6/13 16:33	4
1/6/13 17:10	5.6	Mindanao, Philippines(7.215, 124.839)	Paradise Cagayan De Oro, Philippines(10.69,122.54)	461.577	1/6/13 17:12	1
1/6/13 17:40	4.9	President Roxas, Philippines(7.249, 125.078)	Bukidnon, Philippines(7.9862, 125.0388)	82.091	1/6/13 17:43	2
1/6/13 19:35	4.8	Darapidap, Philippines(17.137, 120.130)	Vigan City, Philippines(17.5704, 120.3873)	55.401	1/6/13 19:39	3

Table 5.9 denotes fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (Min. decision time=0:06, Max. decision time=2:33, Avg. decision time=0:59. Min. location estimation=1.02km, Max. location estimation=67.42km, Avg. location estimation=19.98km)

Table 5.9. Fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (24/5/13 22:50 - 25/5/13 00:00).

Source	**Incident time**	**Actual Location**	**Computed Location**	**ΔLocation (km)**	**Computed Incident time**	**ΔT (HH:mm)**
yes [35]	24/5/13 22:50	Hambley Boulevard, Pikeville, KY 41501, USA(37.482042, -82.5324469)	Hazard, KY, USA(37.24954, -83.1932284)	63.863	24/5/13 23:26	00:36
yes [36]	24/5/13 22:05	Davis Street, Syracuse, NY 13204, USA(43.039217, -76.169803)	Syracuse, NY, USA(43.0481221, -76.1474244)	2.071	24/5/13 23:53	01:48
yes [37]	24/5/13 23:30	Tierra Oaks Drive, Redding, CA 96003, USA(40.6785574, -122.344057)	Redding, CA, USA(40.5865396, -122.3916754)	10.993	25/5/13 00:42	01:12
yes [38]	24/5/13 23:59	506 Wellington Avenue, Cranston, RI 02910, USA(41.7742829, -71.42656)	Lincoln, RI, USA(41.9211111, -71.435)	16.342	25/5/13 00:34	00:35
yes [39]	25/5/13 00:15	1400 Goodview Road, Blue Ridge, VA 24095, USA(37.2651074, -79.7883703)	Bedford, VA, USA(37.3082151, -79.6052884)	16.891	25/5/13 00:37	00:22
yes [40]	25/5/13 00:46	Falterman Road, Duson, LA 70529, USA(30.2272713, -92.2011529)	Lafayette, LA, USA(30.2240897, -92.0198427)	17.424	25/5/13 00:58	00:12
yes [41]	25/5/13 00:00	100 Lazarcheff Drive, Cahokia, IL 62206, USA(38.552003, -90.16671)	Cahokia, IL, USA(38.5708849, -90.1901113)	2.924	25/5/13 01:34	01:34

Table 5.10. Fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (25/5/13 12:00 - 27/5/13 15:43).

Source	**Incident time**	**Actual Location**	**Computed Location**	**ΔLocation (km)**	**Computed Incident time**	**ΔT (HH:mm)**
yes [42]	25/5/13 12:00	6500 Windsor Street, Philadelphia, PA 19142, USA(39.9292686, -75.2411408)	Philadelphia, PA, USA(39.952335, -75.163789)	7.076	25/5/13 14:33	02:33
yes [43]	25/5/13 13:45	Sandalwood Drive, Manassas, VA 20110, USA(38.7424903, -77.4708337)	Manassas, VA, USA(38.7509488, -77.4752667)	1.016	25/5/13 14:36	00:51
yes [44]	25/5/13 15:00	Matthews-Mint Hill Road, NC, USA(35.1476471, -80.6805839)	Matthews, NC, USA(35.1168131, -80.7236804)	5.207	25/5/13 16:54	01:54
yes [45]	25/5/13 16:00	4250 Murray Avenue, Pittsburgh, PA 15217, USA(40.4223167, -79.9295014)	Pittsburgh, PA, USA(40.4406248, -79.9958864)	5.976	25/5/13 16:33	00:33
yes [42]	25/5/13 16:21	Plano Road & Walnut Hill Lane, Dallas, TX 75238, USA(32.8789841, -96.7003836)	Walnut Square Drive, Plano, TX 75025, USA(33.0734892, -96.7364169)	21.888	25/5/13 16:27	00:06
yes [46]	27/5/13 15:43	Glenroyd Gardens, Bournemouth, Dorset BH6 3JN, UK(50.7271337, -1.8022482)	Southbourne, West Sussex PO10, UK(50.844258, -0.906953)	64.272	27/5/13 16:00	00:17

Table 5.11. Fire incident occurrences detected by EmNotifierand verified by trustworthy sources that contain actual time of incident (27/5/13 17:46 - 27/5/13 20:00).

Source	**Incident time**	**Actual Location**	**Computed Location**	**ΔLocation (km)**	**Computed Incident time**	**ΔT (HH:mm)**
yes [47]	27/5/13 17:46	200 Honeysuckle Trail, Wagener, SC 29164, USA(33.6437013, -81.3122098)	Augusta, GA, USA(33.4734978, -82.0105148)	67.417	27/5/13 18:24	00:38
yes [48]	27/5/13 21:00	Manor Park, Woolfardisworthy, Bideford, Devon EX39 5RH, UK(50.9661344, -4.3763572)	Bideford, Devon, UK(51.016684, -4.206666)	13.140	27/5/13 21:43	00:43
yes [49]	27/5/13 20:00	A32, Gosport, Hampshire, UK(50.8123935, -1.1555222)	94A High Street, Gosport, Hampshire PO12 1DS, UK(50.79470828,-1.11960286)	3.200	27/5/13 21:48	01:48

5.4. Implementation related experiments

In this section we have simulated duration for location extraction of a single tweet. Additionally, notification medium delay for Public Wifi, Edge and 3G connection have been experimented.

5.4.1. Processing time of a single tweet

Time spent for location extraction from a tweet directly affects the detection time of an incident. Due to shortage of system memory we were able to use a small size of DBPedia:Place dump that is used for location naming. For this reason we have used MapQuest API to find the GPS coordinates of any location. We have constructed a batch list for geocoding due to limitations of MapQuest system. Locations in the batch

list have been collectively queried for geocoding and reverse geocoding to MapQuest once the list achieved a certain value. Batch list size directly affect the processing time of a single post. For this reason we have experimented the processing time for a single tweet with different batch list sizes. We have developed a simulation environment where tweet frequency is 1 per 2 second. A single tweet might contain at minimum 0 and maximum 4 location indicators. This is the count of fields user can share location information. For instance, a tweet might contain only profile location or profile location, in post location, timezone and attached GPS and their variations. We have suggested that tweets of same number of location indicators arrive repeatedly. According to that, we have measured delays to process a single tweet with different batch sizes (see Table 5.12, 5.13, 5.14).

As it is in real time run, tweet frequency per second directly affects the filling duration of batch list. Besides that, response time of MapQuest system is the other factor for processing time of a single tweet. While the system were running in real time, we have suggested that a single tweet might contain all of the location indicators and tweet frequency is high. According to our experiments, batch size of 5 is the optimal size not to force the limits of MapQuest system.

Table 5.12. Batch size=1. Avg. delay with location count 1=7.293 sec, Avg. delay with location count 2=9.655 sec, Avg. delay with location count 3=32.129 sec, Avg. delay with location count 4=39.191 sec.

	Delay with count of location indicators in post(seconds)			
Test number	**1**	**2**	**3**	**4**
1	7.055	10.764	35.106	41.122
2	5.510	8.403	28.660	40.190
3	8.615	9.937	34.928	41.327
4	7.338	9.068	31.566	41.943
5	7.151	9.407	31.982	37.104
6	6.621	8.348	34.419	41.225
7	8.820	10.779	33.134	35.469
8	7.706	10.118	34.358	38.671
9	7.253	9.447	25.250	39.492
10	6.865	10.280	31.882	35.375

Table 5.13. Batch size=5. Avg. delay with location count 1=15.926 sec, Avg. delay with location count 2=12.825 sec, Avg. delay with location count 3=13.946 sec, Avg. delay with location count 4=9.121 sec.

	Delay with count of location indicators in post(seconds)			
Test number	**1**	**2**	**3**	**4**
1	15.598	12.242	13.443	8.615
2	16.792	12.566	14.964	9.660
3	15.224	13.620	12.819	8.172
4	15.261	12.208	11.654	9.307
5	15.543	12.570	13.399	9.555
6	16.918	12.216	14.282	9.005
7	16.284	13.506	17.382	8.511
8	15.698	12.593	12.94	9.028
9	15.927	12.783	13.673	9.643
10	16.012	13.949	14.902	9.712

Table 5.14. Batch size=10. Avg. delay with location count 1=27.694 sec, Avg. with location count 2=15.113 sec, Avg. delay with location count 3=16.705 sec, Avg. delay with location count 4=12.053 sec.

	Delay with count of location indicators in post(seconds)			
Test number	**1**	**2**	**3**	**4**
1	27.972	16.500	17.467	12.075
2	27.673	15.469	15.013	11.857
3	27.510	13.403	14.064	11.206
4	28.620	15.724	17.525	12.725
5	27.013	16.426	15.604	11.982
6	27.120	15.417	18.459	11.339
7	27.919	14.551	15.321	11.763
8	27.103	14.036	18.921	12.675
9	27.015	16.288	17.010	12.058
10	28.999	13.316	17.667	12.849

5.4.2. Notification medium experiments

We have used push messaging technology for smartphones as notification medium. As discussed in Section 4.2.2 it is not reliable as SMS technology. In this section, we have tested notification delay in Public Wifi, Edge and 3G. As it can be seen from Table 5.15, Public Wifi and 3G users got notification on average 5.407 seconds and 5.477 seconds respectively, after a decision taken by the EmNotifier. This delay includes the response time of APNS as well. Notification have been send to mediums in Table 5.15 at the same time. For this reason we can suggest that the APNS response time is same. As a result we have experimented that Edge users got notification at last. In general we can say that whatever the notification medium is, it has been delivered under 10 seconds with success.

Table 5.15. Push notification delay on Public Wifi, 3G and Edge Average delay Public Wifi=5.407sec, Average delay Edge=7.0545, Average delay 3G=5.477.

Case	**Public Wifi**	**Edge**	**3G**
1	5.543	7.825	5.396
2	5.149	9.013	5.404
3	5.844	5.854	5.002
4	5.036	6.465	5.352
5	5.283	6.125	5.801
6	5.882	8.320	5.789
7	5.014	5.974	5.933
8	5.758	7.784	5.376
9	5.473	6.409	5.239
10	5.087	6.776	5.473

6. DISCUSSION AND FUTURE WORK

In this thesis, we have designed a model for social media based emergency notification. Our model proposes an approach for detecting incident related contributions and if available location extraction from microblog posts. We have chosen Twitter as the source and implemented a real time running prototype of our model. Evaluation of our model basis on data we have fetched from Twitter system. We have tested our model for two types of incidents. These are large scale incident; earthquakes and local incident; fires. We have evaluated our system based on classification performance, NER performance, incidents detected, notification medium delay, processing duration of a single post. Evaluation results indicate that microblogging systems can be utilized as a source for incident detection. We also denote that with limited resources it is possible to deploy a complete real time running notification system with promising results. Section 6.2 discusses about contribution characteristics for fire and earthquake incidents on Twitter. We have depicted several earthquake, fire incident related contribution templates and sample tweets.

6.1. Discussion

There are several topics that we have wondered as a research question. Firstly, does incident related information available in microblogging systems as it is in online newspapers or trustworthy sources? If incident related information is available in microblogging systems what are the similarities and dissimilarities between trustworthy sources? How long does it take for an incident to be published on trustworthy sources versus microblogging systems? In general, what kind of information do contributors share on microblogging systems, are there any statistical text based differences between incident related and unrelated contributions?

Our primary purpose is to generate a notification that would alert people in case of approaching incidents. We have tried to answer if it is feasible to make use of microbloggers contributions for incident detection to generate notification. Umpteen

posts people do share on microblogging systems. Unfortunately, most of them are far from being pertinent about an incident. Information is available for incident detection, but once we perform crosscheck with trustworthy sources, we have observed that the possibility of miss for an incident detection is inevitable.

While testing the system we wanted to experiment if it was able to detect large scale and local incidents. On account of this reason, we have selected earthquake and local fires as test cases.

Human sensors might get corrupted likewise electronic sensors. In that case we are not able to collect incident related information or unfortunately deliver false notifications to users. We have observed that large scale incidents get immediate attention by microbloggers and shared in microblogging systems within several seconds. Local fires similarly get attention by microbloggers but it takes time for contributors to reflect their experience to microblogging systems. Frequency of contribution per earthquake incident is high whereas its period is infrequent. Contributors generally get noticed by earthquakes greater than magnitude 4.5. In case of local fires frequency of contribution per incident is low whereas incident frequency is high. Contributors convey their experiences by enriching them with photos they shoot.

When we crosscheck contributors response time on microblogging systems for earthquake incidents with exact incident time from trustworthy sources, a delay of one or two minutes is available. Delay is about 6 minutes for local fire incidents. In case there is a moving incident (riot, wildfire, hurricane), contributors post indicating possible regions that could get affected by the incident by taking environmental factors (i.e: wind) into account. In accordance with informative posts we have observed, its promising to deploy an early warning system for several types of incidents, based on microblogging contributions.

Incident related microblog post detection is of primary importance for incident detection. Intensive number of incident unrelated posts that have similar templates to incident related posts resulted in misclassification. Other than contribution character-

istics on microblogging systems, system limitations caused misdetection of incidents. Most importantly we are not authorized to fetch 100% of microblog posts but 1%. Query words we have used for fetching posts from microblogging system consists of reasonable incident describing words and their synonyms. Rather than generic query words search space can be narrowed down by making a research on words that best describe an incident.

We get noticed by the fact that deterministic search about the incident using Search API resulted further information about the incident that we couldn't gathered by Streaming API. That means once we detect an incident occurrence further information about the incident can be collected by performing a query with that specific incident related words on Search API.

Regardless of incident type, location information is of primary importance for incident detection. In our study, we have found that contributors share location information of incident in their posts. We have aggregated this valuable information by other location indicators on contributors profile (profile location, timezone, attached GPS) for accurate location estimation. Microblogging users are increasingly getting aware of the fact that their incident related contributions become valuable. In this context, it can be said that contributors started to share informatively about any incident. GPS is the most accurate location information we could gather from a contributors post. However, it doesn't always mean that current location of user is the incident location as a result in post location information come into prominence.

According to our analysis percentage of GPS attached microblog posts is low. However, we expect it to rise with increasing number of GPS supported applications. Mobile application we have developed for incident notification is not something like incident related newspaper. We want people to get alerted by the fact that there is an approaching incident. In this thesis we have focused on location extraction from microblogging posts and notification, albeit microblogging systems are promising systems with human sensors as an early warning system. Information shared by contributors can be enriched for early notification by the facts about an incident characteristics,

such as taking propagation speed of earthquake or speed of wind for fire incidents.

It is important for EmNotifier kind of systems to detect incidents accurately, rapidly, in real-time. Generally we think that such kind of a system should have certain features. As discussed in study by Petrovic [50] these can be listed as;

(i) *Generality.* Many of the developed systems have focused only to detect one type of an event. For example: earthquake, hurricane. The location of the event has been obtained from the users' profiles, timezone or GPS data. The model we have proposed in our study could be applied for any kind of event. Besides we showed that it did obtain the most exact location data from users shared messages. A good system should be able to determine the location of the event as exact as possible for different types of events.

(ii) *Scalability.* In our present life, in emergency situations, people are sharing the events they encountered by social media. Even though, social media systems allow researchers to use only a limited percentage of these messages, obtained unstructured data has a high volume. An emergency system based on social media should handle high volume data and should not slow down as time passes.

(iii) *Real-Time processing.* In emergencies, information regarding the event should be sent to the users as soon as possible. We can't call the systems suitable for scalability metric (able to manipulate high volume data) but not able to find the result in a short time period as real-time. A real time system should be able to process all shared data regarding the event, to make the necessary analyses and to send notifications to the users in a minimum period of time.

(iv) *Classification.* We used supervised classification methods, to determine whether a message belongs to an event or not. If we take generality metric into account, we emphasized that a good system should not only analyze for certain events, it in fact should work for all kinds of events. But we are aware that although such kind of approach is applicable for all events, a good training phase is a must. For developing a training and testing set, we need to annotate manually many messages. Even with a small scale training and testing set, this process will take a long time. No matter how good is the feature selection, a big training set is a

must for the system to work with high accuracy. In this respect, we believe that unsupervised classification methods are more effective and easy to determine the events.

6.2. Contribution characteristics

In order to understand how contributors sharing in case of emergency, we have manually annotated 988 earthquake, 985 fire incident related tweets. This way we try to understand how contributors share location information in their tweets that would result in accurate location extraction for systems using microblogs as a source.

Contributors share location information in post, profile location, timezone or via attached GPS information (see Section 3.13) We have grouped contributors that report about an incident according to location indicators available in their tweets as insider (see Definition 6.1), outsider (see Definition 6.2) and nearby (see Definition 6.3).

Definition 6.1. Insider *A microblogger who mentions about the location of an incident occurrence in post. Additionally, profile location or timezone or GPS information correctly matches with the location of incident occurrence.*

Definition 6.2. Outsider *A microblogger who mentions about the location of an incident occurrence in post. However, profile location or timezone or GPS information does not match with the location of incident occurrence.*

Definition 6.3. Nearby *A microblogger who does not mention about the location of an incident in post. However timestamp of post matches with the timestamp of incident occurrence. Additionally profile location or timezone or GPS information correctly matches with the location of incident occurrence.*

We have examined our dataset for earthquake and fire incidents based upon the definitions (Definition 6.1, 6.2, 6.3). For earthquake incident out of 988 contributors, there exists 189 insider, 139 outsider and 660 nearby reporters. For fire incident out of 985 contributors, there exists 782 insider, 54 outsider and 149 nearby reporters.

One of the crucial difference between insider and outsider can be stated as, outsiders generally retweet the post of insiders. Outsider may have an interest with the place where the incident occurred, but living in a different place. It is also possible that outsider visited a region where an incident broke out, but profile location was not updated. For nearby contributors we can not say they did not experience the incident. In case of earthquake contributors do not give much detail about the location of incident in post. However their profile location overlaps with the location of incident occurrence.

According to our definitions, in fire incident, insider reporter percentage is high. Our insider definition can be supported with the fact that insiders do share photos of the incident they shoot.

Based upon these results we have prioritized in post location information as the highest informative incident occurrence location indicator for a contribution.

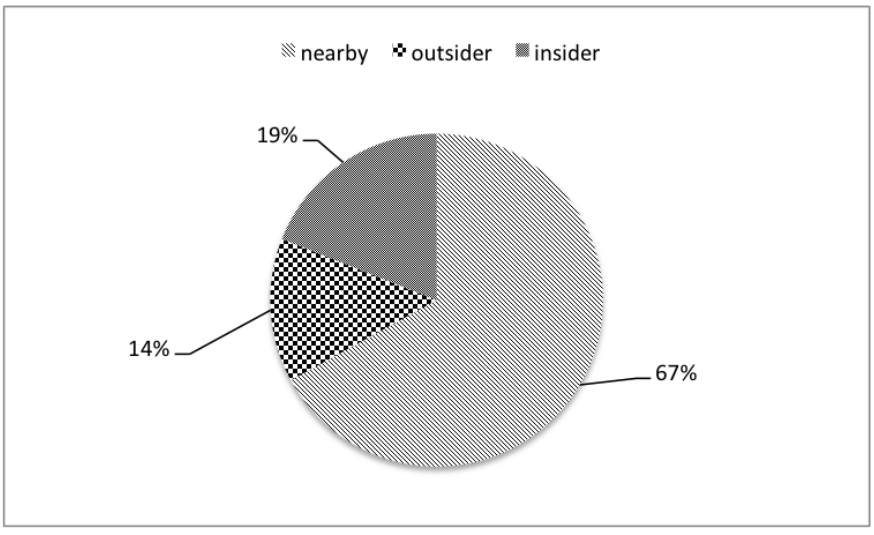


Figure 6.1. Contributors reporting earthquake.

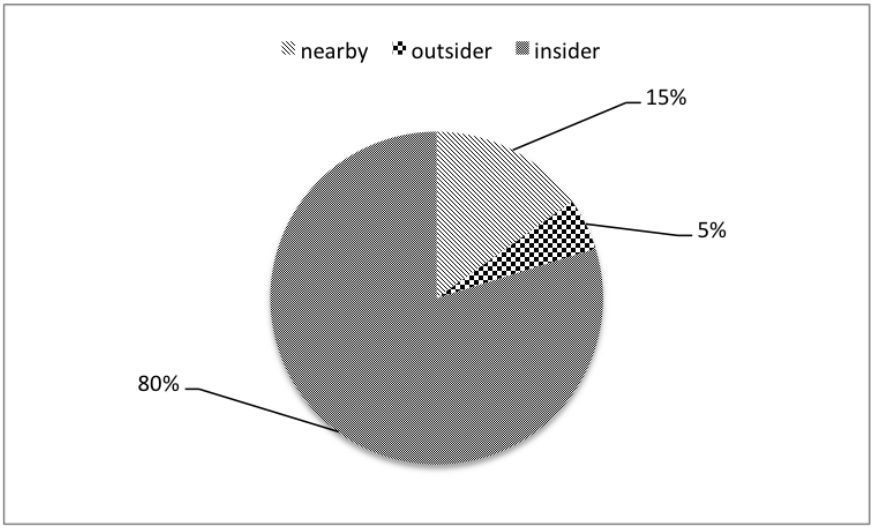


Figure 6.2. Contributors reporting fire.

We have divided contributions as user contribution, official content and negative.

Definition 6.4. (User contribution) *Contributions belong to an incident and do not contain information from trustworthy sources.*

Definition 6.5. (Official content) *Contributions that contain incident related information from trustworthy sources (such as online newspaper).*

Definition 6.6. (Negative) *Contribution that does not belong or mention anything about an incident.*

In order to understand sharing characteristics of contributors, we have manually annotated 810 user contributions, 174 official content tweets for earthquake incident. Additionally, we have manually annotated 912 user contributions, 68 official content tweets about fire incident and total of 30,190 negative tweets.

We have examined basic statistical features of a tweet. These are post length (count of tokens in a post), filtered post length (count of tokens as stop words eliminated but preserved Tag), stop word count, URL percentage, RT percentage, GPS percentage, Mention percentage, Time elements (time describing elements in post such as timestamp or Monday, Tuesday,etc.), in sarcastic (sarcastic location description

such as “the king of white” for “Alaska”), in country (in post country information), in town/state/province/county (in post county, town, state or province information), in city (in post city information), in street (in post street information), in abb (in post location abbreviation), in GPS (in post GPS information), out sarcastic (profile location described sarcastically), out country (profile location given as country information), out town/state/province/county (profile location county, town, state or province information), out city (profile location city sensitivity), out street (profile location street information), out abb (profile location as abbreviation), out GPS (profile location contains GPS information), total location in tweet (percentage of in post location describing elements).

From basic statistical features we can say that user contributions and official content contributions have different contribution characteristics. One of the biggest problem of EmNotifier is to classify user contributions from negative tweets. However, negative contributions found to be containing more stop words than incident related contributions. Stop word count is an important factor to distinguish user contributions from negative tweets.

Contributors sharing about earthquake incident tweet short posts. In post location information of user contributions for earthquake incident is 20%, this value is 94.8% for official content due detailed information about the incident have been included from trustworthy sources.

When we have examined user contributions for fire incident URL percentage is high. Reason is user contributors upload a photo about the incident to an online photo sharing system and embed the link to their posts. In post location indicator percentage is 83.662% which is higher than earthquake incidents.

Taking these basic statistical characteristics into account, our feature vector described in model basis on these factor. We can say it has a slight improvement for classification performance (see Table 5.4, 5.5).

Table 6.1. Tweet characteristics.

Characteristic	Earthquake user positive	Earthquake official content	Fire user positive	Fire official content	Negative
Total tweets	810	174	912	68	30,190
Avg. post length	3	16	17	19	20
Avg. filtered post length	2	13	11	15	7
Stop word count	1	4	6	4	13
URL(%)	1.481	83.908	55.154	58.824	19.089
RT(%)	2.222	10.344	10.964	4.411	26.449
GPS(%)	3.086	31.034	3.179	10.294	1.897
Mention(%)	8.642	15.517	30.372	13.235	46.015
Time(%)	2.347	68.965	2.412	36.764	7.892
In Sarcastic(%)	0.000	0.000	0.000	0.000	2.452
In Country(%)	2.839	47.126	0.548	0.000	2.867
In Town(%)	4.197	32.758	64.803	44.117	4.318
In City(%)	15.555	48.276	15.899	16.176	5.897
In Street(%)	0.124	0.000	12.938	60.294	0.683
In Abb(%)	2.592	6.322	4.166	2.941	6.726
In GPS(%)	0.000	0.000	0.000	0.000	4.863
Out Sarcastic(%)	14.716	4.022	5.043	2.941	28.376
Out Country(%)	22.716	4.597	14.912	35.294	10.654
Out Town(%)	6.790	5.747	45.723	48.529	0.822
Out City(%)	32.098	6.328	35.087	38.235	44.862
Out Street(%)	0.371	0.000	0.438	0.000	1.652
Out Abb(%)	5.309	2.298	22.587	20.588	24.902
Out GPS(%)	4.225	0.578	1.316	4.411	3.871
Total in text location(%)	20.124	94.827	83.662	97.058	5.527

Table D.1, D.2, D.3 provide several contribution templates for earthquake incidents. Table D.4, D.5, D.6 provide several contribution templates for fire incidents. These templates have been extracted based on SVM classification that scored highest confidence in training data.

Sample user positive contributions in case of earthquake, fire, riot, flood, hurricane emergency situations have been given in Table E.1, E.7, E.8, E.4. Sample official content contributions for earthquake and fire incidents are given in Table E.2, E.5. It is difficult for some contributions to determine if they are related to an incident or totally negative contributions. Sample neutral contributions are depicted in Table E.3, E.6.

6.3. Sample incident detections by EmNotifier and related tweets

In this section, we provide a list of several incident occurrences detected by EmNotifier alongside with computed and actual incident location. Words that are bold in the list, depict location entities recognized by EmNotifier. Location extraction from tweet is discussed in Section 3.5.

(i) *Incident information:* 05/11/2012 13:08, 4.8 magnitude, Kathmandu, Nepal (28.411, 86.200)
Computed location: Pathivara Marga, Kathmandu, Nepal, (27.7,85.333)
Δ*Location:* 116.139km

- *Tweet:* Just an Earthquake occurred in *Kathmandu,Nepal*
  Timestamp: 05/11/2012 13:44:58
  Profile Location: Nepal
  Timezone: Kathmandu
  Computed Post location: Kathmandu,Nepal
  Computed Post GPS: 27.7,85.33
- *Tweet: #Kathmandu* #earthquake
  Timestamp: 05/11/2012 13:45:40
  Profile Location: Austria

Timezone: Greenland

Computed Post location: Kathmandu,Nepal

Computed Post GPS: 27.7,85.33

(ii) *Incident information:* 16/06/2012 05:30, East Village Hotel fire, 390 Victoria Street, Singapore 188061 (1.302036, 103.857772)

Computed location: Calgary, Alberta, Canada (51.046, -114.0530)

Δ*Location:* 13661.560km

- *Tweet:* Can see the smoke of *East Village* Hotel fr my office (Revenue House Novena)... `http://t.co/ss71C870`

  Timestamp: 16/07/2012 05:56:51

  Profile Location: Tampines

  Timezone: Singapore

  Computed Post location: Calgary,Alberta,Canada

  Computed Post GPS: 51.046,-114.0530

(iii) *Incident information:* 10/03/2013 19:30, Coach Bus Fire on I-70[46] , Interstate 70, Breezewood, PA, USA, (39.9827989, -78.2775789)

Computed location: All cities US&Canada

- *Tweet:* My bus is literally on fire http://t.co/4Y2Yh6prTd

  Timestamp: 10/03/2013 20:14:33

  Profile Location: DC

  Timezone: Eastern Time (US & Canada)

  Computed Post location: All cities in Eastern Time (US & Canada)

(iv) *Incident information:* 26/01/2013 23:55, Pepperbelly's fire[47] , 849 Texas Street, Fairfield, CA 94533, USA (38.2491558,-122.044)

Computed location: Fairfield City, Salono County, California, US(38.2577, -122.054)

Δ*Location:* 1.290km

- *Tweet:* Fire at Pepperbelly's http://t.co/odNPJF0v

  Timestamp: 26/01/2013 03:20:14

  Timezone: Eastern Time (US & Canada)

  Computed Post location: All cities in Central Time (US&Canada)

[46] `http://goo.gl/v2cUu`

[47] `http://goo.gl/CdgYG`

- *Tweet:* #pepperbellys fire raging http://t.co/up3gCtUO

  Timestamp: 26/01/2013 03:20:37

  Profile Location: Fairfield, California

  Timezone: Pacific Time (US & Canada)

  Computed Post location: Fairfield City, Salano County, California, US

  Computed Post GPS: 38.2577,-122.054

- *Tweet:* #pepperbellys #fire #holyshxt #stillburning http://t.co/BI8Cw1v2

  Timestamp: 26/01/2013 04:59:54

  Profile Location: Fairfield, CA

  Timezone: Pacific Time (US & Canada)

  Computed Post location: Fairfield City, Salano County, California, US

  Computed Post GPS: 38.2577,-122.054

(v) *Incident information:* 05/11/2012 14:00, Sherwood Inn Fire[48] , 19 Murphy Street, Trenton, ON K8V 3X9, Canada (44.1011777,-77.5772114)

Computed location: Ontario, Canada (44.15, -77.25)

Δ*Location:* 26.675km

- *Tweet:* Flames coming from the east side of the Sherwood building #*quinte* #Quinte_Region @QNetNews http://t.co/TRUHwt4D

  Timestamp: 05/11/2012 15:31:02

  Profile Location: Canada

  Timezone: Pacific Time (US & Canada)

  Computed Post location: Ontario, Canada

  Computed Post GPS: 44.15, -77.25

[48] `http://goo.gl/JVAhv`

6.4. Future work

During evaluation we have observed several improvements can be made to our proposed model.

6.4.1. Query expansion

- Query words we have used for fetching posts from microblogging system consists of reasonable incident describing words and their synonyms. According to our analysis (see Section 6.2), we can enrich query words by frequently used co-occurring words in incident related contributions. Such an enrichment might increase the possibility to fetch more incident related posts.
- Rather than generic query words, search space can be narrowed down by making a research on words that best describe an incident.
- Additionally, query expansion libraries can be used such as LucQE.[49]

6.4.2. Disambiguation

- Microbloggers frequently use abbreviations because of character limitation. For instance, "Ca." can be abbreviated by microbloggers as a placement for "California" or "Canada". There can be local descriptions for a place like "Bay area" for "California". In order to effectively extract location from a contribution, a system should be able to handle different type of location entity usages. On the other hand, state codes are frequently used by microbloggers profile or within post. Disambiguation is a necessity to resolve the referenced place.
- Microbloggers might not directly provide the exact name of location but rather refer to that place. Its frequent microbloggers refer to an organization or popular landmarks on the region to be descriptive about exact location of incident. Our model can be enriched by training a named entity recognizer for organization entity detection in sentences, and semantic sources can be used to obtain the location of organization.

[49] `http://lucene-qe.sourceforge.net/`

6.4.3. Performance

We have deployed a local DBPedia server for location extraction. Unfortunately, technical inabilities constrain us to utilize smaller size of DBPedia dump that doesn't contain detailed information about places. We have used Map Quest Open Street Map API for geocoding and reverse-geocoding. Of the use of another system, resolving location of a post caused on delays. Once we utilize LinkedGeoData kind of OpenStreetMap and DBpedia combinations on local server, semantic sources can be used efficiently for faster location resolution.

6.4.4. Incident detection

- Firehose connection with Twitter might be beneficial to figure out incident related contributions of users from a wider perspective, alongside with the possibility to detect more incident occurrences. However such an authorization would not be possible for standard user.
- Search API returned further posts about incident once we have carried out a search with a specific query. Once our system detect an incident candidate it can initialize a search on a different thread specific to incident candidate. Thus more microblog posts can be collected for accurate location estimation using the power of crowd.

6.4.5. Vandalism

Although we have conducted experiments for trust issues based on formulations that take follower and following count into account we found them to be easily failed. One possible approach that we have in mind is to examine the microblogging history of the contributor to design accurate trust model. Currently, our system generate notification based on contributions per incident. In this case, malicious users might come together to false alert the system.

6.4.6. Classification

- We have used supervised learning for user positive and negative post classification. Unfortunately this approach let us to an exhausting and time consuming annotation process of over thousand posts. We have to experiment unsupervised methods to figure out their accuracy. Currently, classification performance is closely related to trained dataset.
- Although microblogging systems utilize their own spamming techniques, in our study everything not related to incidents can be treated as spams. We can experiment term variation or different text characteristics to improve spam filtering.

6.4.7. Handling global incidents

- We have examined only posts that are in english language. However, an emergency system should be able to generate notifications in multiple languages. Herein, contributions on different languages should be examined.
- Emergency notification should be inclusive and customizable. Blind, illiterate or people with different disabilities should be notified in the most appropriate way.
- We have examined that push notification might not be reliable in case of an emergency situation. Permanent internet connection is necessary for users to get notification. Even if there is based upon different communication medium, delays might be unacceptable. It is not a guaranteed way to notify people in case of emergency. A framework such as SMS, have been in use for a long time. In case of emergency situations in addition to push messaging, use of proven systems for message delivery seems to be more appropriate.

6.4.8. Reusable data

Integration of emergency notification systems with emergency response systems would be advantageous. Different systems might communicate with each other using a common standard as RDF. An ontology definition for emergency system would be beneficial and output (incident detected and location) can be generated as RDF.

7. RELATED WORK

7.1. Great disasters and Social Media

There are many different events (mainly disasters) that researchers primarily focused on. These events can be thought of driving forces for the researches in contributing the literature. Tohoku earthquake discussed in [7, 51, 52], occurred in western Pacific Ocean 130 km east of Sendai, Honshu, Japan on March 11, 2011. Initial magnitude 9.0 earthquake and multiple aftershocks afterwards (over five hundred of magnitude 4.5 or greater).[50] Damage to communication equipment, power outages, mobile phone's. People used Twitter, and other social networks for propagating safety information [7]. Chilean earthquake of 2010 discussed in [53] occurred off the coast of central Chile February 27, 2010, magnitude 8.8. After the quake Twitter used to tweet critical information like, alerts, missing people, available-interrupted services, road conditions, functioning gas stations. Hashtag #terremotochile (chileearthquake) used frequently, however internet service was nonfunctional after 48 hours in the effected region [53]. Sichuan earthquake discussed in [54] occurred May 12, 2008 China, magnitude 8.0. Millions of people became homeless, thousands of people injured. Web used as a place for sharing information, expressing feeling and opinion [54]. Haitian disaster discussed in [55, 56] occurred January 12, 2010, Haiti, magnitude 7.0 and 52 aftershocks. Three million people affected by the earthquake and 316,000 people died. Major damage in the country, on communication systems, air, land, sea transport, hospitals and electrical networks.[51] Oklahoma grassfires discussed in [55, 57] began on April 9, 2009, Oklahoma. Fire reduced following day but at least 60 injury reports. 270 buildings were destroyed [57]. Magnitude 8.0 earthquake, Pacific near American Samoa, September 29, 2009 triggered tsunami resulted death of 22 people.[52] Red River Flood discussed in [57, 58] crested in Fargo March 28, 2009, flooding remained for several weeks [57]. Icelandic volcano eruption (Eyjafjallajokull) discussed in [59], caused

[50] http://earthquake.usgs.gov/earthquakes/eqarchives/poster/2011/20110311.php

[51] http://en.wikipedia.org/wiki/2010_Haiti_earthquake

[52] http://en.wikipedia.org/wiki/2009_Samoa_earthquake

disruption to air travel across western and northern Europe for six weeks starting by April 2010.[53]

7.2. Information about usage of Social Media in an Emergency Situation

American Red Cross conducted a great online survey about the usage of social media in disasters and emergencies in 2010 [60]. According to their survey Facebook is found to be the most popular social media application. Nearly 75% of people found to be participating in at least one online community. In general when we look at crowdsourcing applications Twitter has been used effectively, which reflects 15% people contributed to this survey. User contribution is another important factor in crowdsourcing applications. There won't be any data to process if no one is contributing to social media. It has been found that 82% of social media users participate actively in social media. 48% of users participate to those systems everyday.

16% of people use social media to get information about emergency situation such as earthquake, flood, tornado etc. [60]. TV news (63%) is the leader medium that people use to get emergency related information. Twitter and mobile application usage are nearly 7%. Encouraging people to participate into social media needs to be one of the primary purposes for researchers in this field. As the amount of data increases more accurate results can be achieved by the developed systems and people than can be satisfied by the results of the system. Good systems might trigger people to contribute to social media and let them to use mobile applications more frequently to get information on emergency situations. About 50% percent of participants responded positively to receive notifications related to any emergency information. This is one of the motivation for us and many other researchers to study more on emergency systems. In order to collect data related to emergency or detect emergency situation, we need people provide information to social media systems. According to the survey, 50% of the participants mentioned about emergency situation in their social media contributions.

[53] http://en.wikipedia.org/wiki/2010_eruptions_of_Eyjafjallajokull

Facebook is the leader social media system for people to share emergency information (75%), followed by blogs (22%) and Twitter (21%). Users who provide valuable information to social media systems, expect emergency responders to monitor or respond to their valuable contributions. User might post a request for urgent help or question verify information going on public about an emergency event. Younger people found to be participating to social media more frequently (89% participate in online communities). Younger participants are likely to mention about emergency situations online and request help through social media (nearly 60%).

Survey that the American Red cross conducted shows us the importance and usage of social media in emergency situation. Although the usage percentages of social media in emergency situations is not so high, it's future is promising for usage in emergency situations. Currently, Twitter or other microblogging applications are commonly used as crowdsourcing environments due their support for user generated data through APIs they provide. Even though character limitation is a challenge in microblogging systems, it has some advantages like processing less amount of textual data. Depending on the context of research, it might be easier to structure microblogging data. Nevertheless, survey depicts Facebook has great usage potential by users in emergency situations. This shows that researchers may focus more on using Facebook data for developing crowdsourcing applications.

7.3. Academic Studies

In [61], Serdyukov *et al.* developed a system for automatically mapping Flickr photos based on tags. They define a matrix representing world coordinates to map the Flickr photo. The input to their system is Flickr photo tag set as respective photo descriptions. Output, estimated location of photo, is predicted according to descriptive tags of photo. Their system relies only on tags of photo, not the technical specification of the photo itself i.e: EXIF data. Location disambiguation and toponym resolution is the main task of their proposed model. Tag set have been processed with a language model to find possible locations that are mentioned by a similar tag set. In case there is no direct disambiguators in tag set, for example country "France" or "United States"

to disambiguate “Paris”, process of spatial space minimalization have been conducted by searching nearby places, i.e: “Nice”. They evaluate their system by measuring the accuracy of correct location prediction. In our proposed model, we do not try to find similar set of text for location estimation. We use named entity recognizer for detection of location descriptors, validate and disambiguate the location using semantic sources. We do not rely only on text itself for disambiguating location extracted from tweet text. Additional sources like the profile location, timezone, and if available GPS position attached to tweet are used. However, Flickr and Twitter are different systems, and provide developers different functionalities. Authors, Serdyukov *et al.* point out the challenges of location estimation from tag set, due tagging behaviour of users. Incorrect tagging, insufficient number of tags to represent photo (lack of disambiguators), generic tagging (tagging “France” rather than “Paris,France” for “Eiffel tower” photo). In case of Twitter, it is highly expected that users provide incorrect additional information like the profile location, or timezone. As we have mentioned in Section 6 generic location description is a common problem. However, as the mobile technology progress, sharing of GPS information increases.

Similar to [61], Rattenbury *et al.* studied semantics of tags to extract event-place information from Flickr tags [62]. Purpose of their research is to find spatial and temporal patterns in some specific domain to identify event-place from tags. The idea is about users contributing specific tags in times where a local event is going to happen. For example, expecting local festival related tags in a specific region at the end of July. Their methodology basis not only tags, but any textual data that contains spatial and temporal information. They define event and place tags as two different concepts. Event tag can be identified by temporal patterns whereas place tags by spatial patterns. They emphasize that, some tags may be used in specific regions in specific times. For example“New York marathon” tag can break out in some specific time and specific location whereas “dog” tag may appear anytime everywhere. Tag usage may also be predefined by geographic information. “Carnival” tag may not appear as frequent as in Brazil in any other region in the world. Intuitive idea of their methodology is splitting time/space into small parts. If some tag in the set has higher occurrence than general occurrence of the same tag in set, it would mean that

there is some event going on at that time. Evaluation of their system is conducted by standard information retrieval methods. They annotate a tag set as events, places and compare how their system identify events, places correctly from a tag set. Sensor based monitoring of any problem generate time-series data. similar to [62] in [63], Guralnik *et al.* studied identifying time periods at which rapid changes occur, called change-point detection. Batch and incremental version of change-point detection was mentioned by their work. Best set of change points from a dataset was determined in the batch version. In incremental version, data is collected one by one in real time manner, and each point is tested to see if it causes a change point. Incident keywords are generic and can be used to express many other things in addition to incidents. The count of tweets that contain the word "fire" in Twitter for a 60 minutes interval can be seen from Figure 7.1. As it is depicted in Figure 7.1 there are 4 local fire incidents during 08:20-08:26 period, but as it is expected there is no peak detect anomalies.

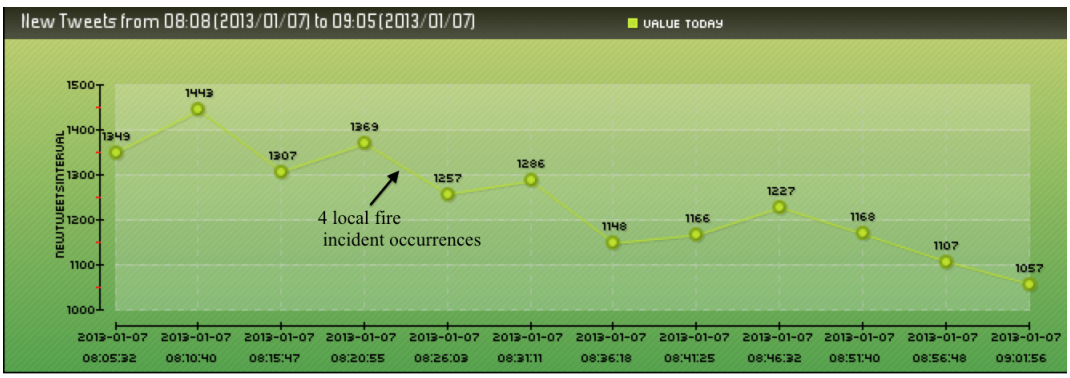


Figure 7.1. Count of "fire" query word related posts from microblogging system in 5 minutes interval.

In [51], Doan *et al.* analyzed tweets in period of Tohoku earthquake in Japan to track awareness levels on social media about earthquake. Tweets that they have analyzed mostly contains contributor expressions about the earthquake incident. They search through Twitter for various keywords to understand public mood. Their research shows that people generally share an event they have experienced right after incident happens. In case of Tohoku earthquake first tweet was after 1 minutes. According to

their research, users tweeted in their native languages at the first seconds of the incident. First message was Japanese, not English. Number of tweets about the earthquake reached peak the next day after the earthquake. We do not expect a peak right after incident, that is because incident related keywords are generic. However, after some time, more information about the incident is begin to flow. As we have found in our research, people share an event immediately after different incidents (see Table 5.6). Unfortunately, our system basis only on English messages, thus we were unable to catch most of the incidents originated from Japan, where earthquake incident is frequent.

In [64], Sutton *et al.* discusses how Twitter was effectively used in Tennessee disaster. Mass media ignored the existence of disaster, however, after a while by the contributions of Twitter users, mass media started to make news about the disaster. This shows that Twitter is a powerful medium that is used frequently by people in case of emergency. This work by Sutton*et al.* perfectly fits to our motivational idea that, there might be local incidents (local fire, accident, etc.) that mass media do not cover. Incident in Tennessee points out to a great real life example about ignorance of incidents by mass media. Trust has crucial importance in emergency systems. Author also analyzed Twitter posts within a time period, to find identities of contributors. That is to understand if locals affected by the disaster or sensitive people outside the affected region contribute more to social media.

In [58], Palen *et al.* analyzed tweets at the times of Red River flood in 2009. They try to find out profile information of users (local people contribute to social media or people outside the affected region) that contributed to social awareness. In order to achieve their goal, they have annotated user location information on a map. According to their research, people outside the affected region contributed more about the event. In the literature there are studies to calculate earthquake epicenter based on user profiles of Twitter. A system that purely basis on profile location for location estimation might perform poorly because of the fact that people living around the epicenter would be affected more due to earthquake, so we might get less user contribution from that particular location. Unfortunately such an assumption might result in inaccurate location estimation, because of the fact that contributors outside

the affected region might have profile locations pointing to different regions. Briefly according to their research it can be said that tweeting behaviour changes based on the distance of the user to disaster. While locals tweet informative messages, people who are not affected by the disaster generally retweet messages they found to be informative. People generally trust local media, or organizations in order to be updated about the incident. Currently, information propagates rapidly through social media in different forms, as a result retweets got great attention in emergency situations. People generally use retweeting in order to notify others that something important is going on. Their findings show that 10% of the emergency related tweets are sent by users that are local to event. Local people in the disaster region tweet more regional and informative messages. According to Palen *et al.* noise while collecting emergency related tweets can be reduced by conducting keyword based searches and using retweet messages. Retweet count will show us the value of that information in case of emergency. According to their research, retweets contain more headline messages, whereas detailed, informative messages generally are posted by locals. In our proposed model, we have used valuable in post location indicators as well as contributors profile location for location estimation (see Section 6.2 for our definition of insider, outsider).

In [53], Mendoza *et al.* explore the behaviour of Twitter users under emergency situations. They analyze how rumours propagate in Twitter network and its possible detection strategy. Their findings show that rumours propagate differently than confirmed/correct news. Rumours are generally more questioned (RT's with question comments) so detection of rumours can be possible. According to the research, institutional accounts tweet more in case of disaster. In our proposed model, one of the reasons why we have included official content tweets is because there might be some incidents where there is no user contribution. However, institutional accounts are active and they generally tweet about the incident as soon as the incident happens. Of course the primary source of our model is user contributions but official content tweets about an incident cannot be ignored. Retweet's help propagation of information and the number of RT's for a tweet shows the importance of tweet. Mendoza *et al.* inspect the vocabulary of tweets in emergency situation, they show frequently used words in a cloud [53]. They categorized tweets in 5 categories; affirms (tweets are trustworthy),

denies, questions, unrelated, unknown. If some tweet is correct (after conforming tweet by other sources) deny amount of it is less. If information is incorrect than denial of tweet is high. By comparing questionable tweets with confirmed truths it has been found that questionable tweets generally provide incorrect information. This shows us that rumours are to be questioned more.

In [57], Vieweg *et al.* analyze Twitter messages in emergency events Oklahoma grassfires 2009 and Red River floods 2009. They are trying to identify type of messages that contribute to situational awareness. In case of emergencies, researchers must decide rapidly on what information to focus and how to collect data. They suggest that the first step to collect data starts by searching tweets with relevant keywords about the incident. This would produce less noisy dataset. Tweets in their dataset are divided by manual annotation into two as; tweets relevant to emergency (on topic) and off topic. Contributors are then grouped as individuals, organization and locals. They infer location information from user profile or tweets manually. According to their analysis geo location information is generally shared in tweets that can be used for automatic extraction of tweets. In Oklahoma dataset 40% of tweets contain geo location information and 18% in RedRiver dataset. Location information is generally found to be shared, if users are able to guess the next probable direction of the event. In case of fire, people share the location of the fire by taking wind into account. So, one might directly infer that fire is getting closer to his/her land. But in case of flood, speed of propagation is faster and direction of flood cannot be estimated clearly. We can say that contributors are aware that location information is crucial in case of an emergency. Precise location information is generally shared within tweets, if users are able to estimate the current event location. Authors also discuss about location referencing. Twitterer might not know the exact location, but can point the location by referencing to some other location. Indication of location information might be useful for extracting exact location for events, but not studied well on literature for now. In our research, we have analyzed tweets in the same manner and expose that location information is shared by contributors in case of emergency situation. We further research on this topic and propose a model about extracting location information from tweet.

In addition to location information, Vieweg *et al.* note that advise type of messages are found to be popular in early hours of events. After the affects of incident begin to reduce, volunteer information found to be propagating. They have found that high yield contributors tweet informative, structured (name, geolocation, time, etc) information. These type of contributors have high number of followers and they are aware of their public role. Authors mention that tweets that contain geolocation and situational updates are retweeted more than lack of geolocation attached tweets. Markedness is another factor they have encountered while they were analyzing tweets. Rather than abbreviations, people might tweet "river" rather than "red river" because of the fact that "red river" is popular at that time and people can immediately infer "red river".

[52, 65, 66] focus on earthquake incident detection using Twitter. Similar to our proposed model Okazaki *et al.* and Sakaki *et al.* monitor tweets and detect events in real time using supervised machine learning approaches. They have prepared training data by annotating 597 positive tweets. SVM have been used for classifying tweets into positive and negative classes. Their feature vector consists of simple statistical features such as count of words in a tweet, position of query word, words in a tweet. Overall their system detect earthquakes that are bigger than magnitude 3 by 93%. Although our classification methodology is similar to theirs, we take in tweet text location, special features of Twitter (hashtags, emoticons, punctuations, URLs) as additional features. In Section 6.2, we have showed that statistical characteristics of Twitter contributions change according to incident type. It shows that our model that is influenced by Sakaki *et al.* can be used for classifying different types of incidents. Rather than simply focusing on profile location for location estimation, taking into the fact that currently people get started to share location information in tweet text, more accurate location estimation can be done for incidents. They have evaluated their system by comparing detected earthquake incidents with USGS, classification performance with features used. In [66], Earle *et al.* detect earthquake events by change point detection method. They construct tweet frequency time series and detect the peak point, meaning there is possibly an incident. They collected 5 months of earthquake related Twitter data and able to detect 48 earthquakes over 2 magnitude. At the same period they note

that U.S Geological Survey reported 5175 quakes. Detected earthquakes were the ones that were widely felt events and shared frequently on Twitter. In some less populated regions due earthquake information was not shared through Twitter, seismographic detection was found to be faster. Their method looks for rapid increase in tweets that contain a keyword such as "earthquake". For detecting the locations of the earthquakes, they use GPS location attached to tweets or location information provided by users profile. Their approach to incident detection might be valid for earthquake incidents. That is earthquake incidents are generally felt across multiple regions and by hundreds of people. However, local events like fire, traffic accidents does not result in a peak to detect by change point detection method.

Ushahidi [67], is well known crowdsourcing application. The purpose of Ushahidi is to benefit from crowd to facilitate crisis information sharing. As Okolloh discusses in [67], project started to be developed in Kenya at the time of violence after election in 2007. Their open source application provides information collected (from emails, SMS, crowdsourcing social media, direct reporting to system), analysis of information and mapping. Their system can handle various types of incidents such as violence reports at a specific location, flood, terrorism, fire. Methodology of their system is not given in detail, so we are not able to discuss about their methodology on how they analyze information.

Trust is one of the most important issue in crowdsourcing projects. Ushahidi have solved this problem by employing paid/volunteer local sources. Local sources are trusted and can identify if reports at that region is hoax or not. Having many sources mean more reliable crisis information. All of the reports are verified by local sources, but the number of reports are also important as a sign of trusted information. For example, a spammer hoaxes about a specific region, violence in region "X". Many other sources may report in contrary about this hoax report. As Okolloh says, erroneous information be identified by media check. However, there might be incidents that are not covered by mass media.

In [68, 69], Abel *et al.* developed system named "Twitcident". It uses emer-

gency broadcasting systems to detect incidents and get detailed location information. It then starts to track incident related information on Social Media. As an emergency broadcasting system they use P2000 pager device and transform the incoming message to query for collecting incident related information on Twitter. They have extracted location information from pager message using regular expressions. Emergency broadcasting device gives them great detail of location information about the incident. However, different emergency services generate various types of messages that might reduce accuracy of location extraction using regular expression. Most of the challenge of their work is on query enhancing about incident. Our proposed model solely basis on location extraction from Social Media. We show that with enough number of contributions and by using valuable in tweet text location information by contributors, more precise location estimation can be done.

8. CONCLUSION

In this thesis, we have exposed how microblogging systems are effectively used in case of emergency situations by providing examples from fire, earthquake, riot and hurricane. We have studied if microblogging systems contain information about location and timestamp of any incident that would allow researchers to detect an incident occurrence. We have chosen fire and earthquake related incidents to figure out if location information is available on microblogging contributions. As a result, we have determined that valuable location information to detect the occurrence of an incident is available on microbloggers contributions. Unfortunately, location sensitivity that is shared by contributors is low. Location information about the country and city can be found however, sharing of high sensitive location information is low.

Nowadays, number of GPS aided devices and applications are increasing. We suggest that in future, contributions will contain more higher sensitive location information.

We have proposed a model to analyse contributions on microblogging systems, determine the presence, location of incident and notify interested people. We have developed a prototype of our model that is running in real time and uses Twitter as microblogging system.

Developed prototype detects fire or earthquake incidents from microblogging contributions, estimates the incident location and warn interested users through mobile application. Main purpose of this notification is to warn people as soon as possible before the incident reaches to their location using human sensors. Additionally, people get notification about local incidents that are generally not shared by mainstream media.

APPENDIX A: STOP WORDS LIST

a | able | about | above | abst | accordance | according | across | act | actually | added | adj | affected | affecting | affects | after | afterwards | again | against | ah | all | almost | alone | along | already | also | although | always | am | among | amongst | an | and | announce | another | any | anybody | anyhow | anymore | anyone | anything | anyway | anyways | anywhere | apparently | approximately | are | aren | arent | arise | around | as | aside | ask | asking | at | auth | available | away | awfully | b | back | be | became | because | become | becomes | becoming | been | before | beforehand | begin | beginning | beginnings | begins | behind | being | believe | below | beside | besides | between | beyond | biol | both | brief | briefly | but | by | c | ca | came | can | cannot | can't | cause | causes | certain | certainly | co | com | come | comes | contain | containing | contains | could | couldnt | d | date | did | didn't | different | do | does | doesn't | doing | done | don't | down | downwards | due | during | e | each | ed | edu | effect | eg | eight | eighty | either | else | elsewhere | end | ending | enough | especially | et | et-al | etc | even | ever | every | everybody | everyone | everything | everywhere | ex | except | f | far | few | ff | fifth | first | five | fix | followed | following | follows | for | former | formerly | forth | found | four | from | further | furthermore | g | gave | get | gets | getting | give | given | gives | giving | go | goes | gone | got | gotten | h | had | happens | hardly | has | hasn't | have | haven't | having | he | hed | hence | her | here | hereafter | hereby | herein | heres | hereupon | hers | herself | hes | hi | hid | him | himself | his | hither | home | how | howbeit | however | hundred | i | id | ie | if | i'll | im | immediate | immediately | importance | important | in | inc | indeed | index | information | instead | into | invention | inward | is | isn't | it | itd | it'll | its | itself | i've | j | just | k | keep | keeps | kept | kg | km | know | known | knows | l | largely | last | lately | later | latter | latterly | least | less | lest | let | lets | like | liked | likely | line | little | 'll | look | looking | looks | ltd | m | made | mainly | make | makes | many | may | maybe | me | mean | means | meantime | meanwhile | merely | mg | might | million | miss | ml | more | moreover | most | mostly | mr | mrs | much | mug | must | my | myself | n | na | name | namely | nay | nd | near | nearly | necessarily | necessary | need | needs | neither | never | nevertheless | new | next

| nine | ninety | no | nobody | non | none | nonetheless | noone | nor | normally | nos | not | noted | nothing | now | nowhere | o | obtain | obtained | obviously | of | off | often | oh | ok | okay | old | omitted | on | once | one | ones | only | onto | or | ord | other | others | otherwise | ought | our | ours | ourselves | out | outside | over | overall | owing | own | p | page | pages | part | particular | particularly | past | per | perhaps | placed | please | plus | poorly | possible | possibly | potentially | pp | predominantly | present | previously | primarily | probably | promptly | proud | provides | put | q | que | quickly | quite | qv | r | ran | rather | rd | re | readily | really | recent | recently | ref | refs | regarding | regardless | regards | related | relatively | research | respectively | resulted | resulting | results | right | run | s | said | same | saw | say | saying | says | sec | section | see | seeing | seem | seemed | seeming | seems | seen | self | selves | sent | seven | several | shall | she | shed | she'll | shes | should | shouldn't | show | showed | shown | showns | shows | significant | significantly | similar | similarly | since | six | slightly | so | some | somebody | somehow | someone | somethan | something | sometime | sometimes | somewhat | somewhere | soon | sorry | specifically | specified | specify | specifying | still | stop | strongly | sub | substantially | successfully | such | sufficiently | suggest | sup | sure | t | take | taken | taking | tell | tends | th | than | thank | thanks | thanx | that | that'll | thats | that've | the | their | theirs | them | themselves | then | thence | there | thereafter | thereby | thered | therefore | therein | there'll | thereof | therere | theres | thereto | thereupon | there've | these | they | theyd | they'll | theyre | they've | think | this | those | thou | though | thoughh | thousand | throug | through | throughout | thru | thus | til | tip | to | together | too | took | toward | towards | tried | tries | truly | try | trying | ts | twice | two | u | un | under | unfortunately | unless | unlike | unlikely | until | unto | up | upon | ups | us | use | used | useful | usefully | usefulness | uses | using | usually | v | value | various | 've | very | via | viz | vol | vols | vs | w | want | wants | was | wasn't | way | we | wed | welcome | we'll | went | were | weren't | we've | what | whatever | what'll | whats | when | whence | whenever | where | whereafter | whereas | whereby | wherein | wheres | whereupon | wherever | whether | which | while | whim | whither | who | whod | whoever | whole | who'll | whom | whomever | whos | whose | why | widely | will | willing | wish | with | within | without | won't

| words | world | would | wouldn’t | www | x | y | yes | yet | you | youd | you’ll | your | youre | yours | yourself | yourselves | you’ve | z | zero | rt | http | t.co | aren’t | couldn’t | hadn’t | he’d | he’ll | he’s | here’s | how’s | i’d | i’m | it’s | let’s | mustn’t | ours | shan’t | she’d | she’s | that’s | there’s | they’d | they’re | we’d | we’re | what’s | when’s | where’s | who’s | why’s | you’d | you’re | a’s | ain’t | allow | allows | apart | appear | appreciate | appropriate | associated | best | better | c’mon | c’s | cant | changes | clearly | concerning | consequently | consider | considering | corresponding | course | currently | definitely | described | despite | entirely | exactly | example | going | greetings | hello | help | hopefully | ignored | inasmuch | indicate | indicated | indicates | inner | insofar | it’d | novel | presumably | reasonably | second | secondly | sensible | serious | seriously | t’s | third | thorough | thoroughly | three | uucp | well | wonder | I | a | about | an | are | as | at | be | by | com | for | in | is | it | of | on | or | the | to | was | what | who | will | amoungst | amount | bill | bottom | call | computer | con | cry | de | describe | detail | eleven | empty | fifteen | fify | fill | find | forty | front | full | hasnt | interest | mill | mine | move | side | sincere | sixty

APPENDIX B: MICROBLOG POST CLASSIFICATION PERFORMANCE

Earthquake and fire incident related microblog post classification performance with 2, 5, 10 fold cross validation using SVM and Naive Bayes classifer.

Table B.1. 2-fold cross validation, Train set count= 619, Test set count= 619, Positive tweets=396, Negative tweets=223,Min F-Score=0.839, Max F-Score=0.914, Avg. F-Score=0.877.

Fold	**Precision**	**Recall**	**F-score**
1	0.727	0.992	0.839
2	0.874	0.959	0.914

Table B.2. 2-fold cross validation using Naive Bayes classifer. Train set count= 619, Test set count= 619, Positive tweets=396, Negative tweets=223, Min F-Score=0.850, Max F-Score=0.865, Avg. F-Score=0.856.

Fold	**Precision**	**Recall**	**F-score**
1	0.741	0.997	0.850
2	0.824	0.911	0.865

Table B.3. 5-fold cross validation using SVM classifer. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93,Min F-Score=0.874, Max F-Score=0.907, Avg. F-Score=0.892.

Fold	**Precision**	**Recall**	**F-score**
1	0.868	0.95	0.907
2	0.842	0.937	0.887
3	0.849	0.95	0.896
4	0.857	0.937	0.895
5	0.856	0.893	0.874

Table B.4. 5-fold cross validation using Naive Bayes classifer. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93, Min F-Score=0.837, Max F-Score=0.889, Avg. F-Score=0.860.

Fold	**Precision**	**Recall**	**F-score**
1	0.851	0.931	0.889
2	0.812	0.918	0.862
3	0.831	0.893	0.861
4	0.837	0.868	0.852
5	0.820	0.856	0.837

Table B.5. 10-fold cross validation using SVM classifer. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Min F-Score=0.837, Max F-Score=0.939, Avg. F-Score=0.899.

Fold	**Precision**	**Recall**	**F-score**
1	0.935	0.901	0.918
2	0.927	0.950	0.939
3	0.851	0.987	0.914
4	0.894	0.938	0.915
5	0.866	0.962	0.912
6	0.791	0.888	0.837
7	0.872	0.925	0.898
8	0.853	0.938	0.894
9	0.829	0.962	0.891
10	0.848	0.901	0.874

Table B.6. 10-fold cross validation using Naive Bayes classifer. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Min F-Score=0.797, Max F-Score=0.913, Avg. F-Score=0.863.

Fold	**Precision**	**Recall**	**F-score**
1	0.866	0.802	0.833
2	0.875	0.864	0.869
3	0.782	0.975	0.868
4	0.822	0.913	0.865
5	0.813	0.913	0.860
6	0.75	0.851	0.797
7	0.858	0.975	0.913
8	0.848	0.901	0.874
9	0.831	0.913	0.870
10	0.858	0.901	0.879

Table B.7. 2-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 619, Test set count= 619, Positive tweets=386, Negative tweets=223, Avg. F-Score set-1=0.906, Avg. F-Score set-2=0.877.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.907	0.839
2	0.904	0.914

Table B.8. 2-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 619, Test set count= 619, Positive tweets=386, Negative tweets=223, Avg. F-Score set-1=0.858, Avg. F-Score set-2=0.837.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.850	0.840
2	0.865	0.834

Table B.9. 5-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93, Avg. F-Score set-1=0.892, Avg. F-Score set-2=0.824.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.907	0.850
2	0.887	0.793
3	0.896	0.840
4	0.895	0.817
5	0.874	0.822

Table B.10. 5-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 1012, Test set count= 253, Positive tweets=160, Negative tweets=93, Avg. F-Score set-1=0.860, Avg. F-Score set-2=0.825.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.889	0.850
2	0.862	0.793
3	0.861	0.840
4	0.852	0.817
5	0.837	0.822

Table B.11. 10-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Avg. F-Score set-1=0.910, Avg. F-Score set-2=0.899.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.915	0.918
2	0.906	0.939
3	0.938	0.914
4	0.907	0.915
5	0.918	0.912
6	0.884	0.837
7	0.920	0.898
8	0.891	0.894
9	0.899	0.891
10	0.925	0.874

Table B.12. 10-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 1170, Test set count= 130, Positive tweets=81, Negative tweets=49, Avg. F-Score set-1=0.863, Avg. F-Score set-2=0.808.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.833	0.842
2	0.869	0.822
3	0.868	0.839
4	0.865	0.802
5	0.860	0.830
6	0.797	0.793
7	0.913	0.702
8	0.874	0.818
9	0.870	0.824
10	0.879	0.809

Table B.13. 2-fold cross validation using SVM classifer with feature set-1 andfeature set-2. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Avg. F-Score set1=0.882, Avg. F-Score set2=0.834.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.880	0.825
2	0.884	0.844

Table B.14. 2-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Avg. F-Score set-1=0.866, Avg. F-Score set-2=0.852.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.851	0.856
2	0.880	0.848

Table B.15. 5-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Avg. F-Score set-1=0.895, Avg. F-Score set-2=0.845.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.866	0.836
2	0.905	0.845
3	0.906	0.822
4	0.887	0.874
5	0.912	0.848

Table B.16. 5-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Avg. F-Score set-1=0.842, Avg. F-Score set-2=0.831.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.849	0.829
2	0.844	0.815
3	0.791	0.827
4	0.828	0.834
5	0.899	0.852

Table B.17. 10-fold cross validation using SVM classifer with feature set-1, 2. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Avg. F-Score set-1=0.861, Avg. F-Score set-2=0.821.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.856	0.789
2	0.839	0.833
3	0.890	0.816
4	0.828	0.853
5	0.888	0.818
6	0.897	0.841
7	0.850	0.813
8	0.855	0.791
9	0.861	0.839
10	0.846	0.813

Table B.18. 10-fold cross validation using Naive Bayes classifer with feature set-1, 2. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Avg. F-Score set-1=0.816, Avg. F-Score set-2=0.802.

Fold	**F-score feature set-1**	**F-score feature set-2**
1	0.791	0.770
2	0.785	0.797
3	0.779	0.802
4	0.867	0.808
5	0.834	0.825
6	0.770	0.777
7	0.851	0.805
8	0.863	0.814
9	0.819	0.846
10	0.800	0.774

Table B.19. 2-fold cross validation using SVM classifer. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Min F-Score=0.880, Max F-Score=0.884, Avg. F-Score=0.882.

Fold	**Precision**	**Recall**	**F-score**
1	0.871	0.889	0.880
2	0.854	0.916	0.884

Table B.20. 2-fold cross validation using Naive Bayes classifer. Train set count= 407, Test set count= 407, Positive tweets=289, Negative tweets=108, Min F-Score=0.851, Max F-Score=0.880, Avg. F-Score=0.866.

Fold	**Precision**	**Recall**	**F-score**
1	0.870	0.833	0.851
2	0.837	0.927	0.880

Table B.21. 5-fold cross validation using SVM classifer. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Min F-Score=0.866, Max F-Score=0.912, Avg. F-Score=0.865.

Fold	**Precision**	**Recall**	**F-score**
1	0.878	0.855	0.866
2	0.913	0.898	0.905
3	0.874	0.940	0.906
4	0.876	0.898	0.887
5	0.900	0.923	0.912

Table B.22. 5-fold cross validation using Naive Bayes classifer. Train set count= 676, Test set count= 169, Positive tweets=118, Negative tweets=51, Min F-Score=0.815, Max F-Score=0.852, Avg. F-Score=0.831.

Fold	**Precision**	**Recall**	**F-score**
1	0.836	0.822	0.829
2	0.826	0.805	0.815
3	0.842	0.813	0.827
4	0.814	0.855	0.834
5	0.825	0.881	0.852

Table B.23. 10-fold cross validation using SVM classifer. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Min F-Score=0.828, Max F-Score=0.897, Avg. F-Score=0.862.

Fold	**Precision**	**Recall**	**F-score**
1	0.840	0.892	0.865
2	0.833	0.846	0.839
3	0.847	0.938	0.890
4	0.773	0.892	0.828
5	0.857	0.923	0.888
6	0.859	0.938	0.897
7	0.826	0.876	0.850
8	0.808	0.907	0.855
9	0.819	0.907	0.861
10	0.805	0.892	0.846

Table B.24. 10-fold cross validation using Naive Bayes classifer. Train set count= 918, Test set count= 102, Positive tweets=65, Negative tweets=37, Min F-Score=0.770, Max F-Score=0.846, Avg. F-Score=0.802.

Fold	**Precision**	**Recall**	**F-score**
1	0.742	0.800	0.770
2	0.730	0.876	0.797
3	0.719	0.907	0.802
4	0.774	0.846	0.808
5	0.756	0.907	0.825
6	0.708	0.861	0.777
7	0.756	0.861	0.805
8	0.785	0.846	0.814
9	0.805	0.892	0.846
10	0.714	0.846	0.774

APPENDIX C: WORD CLOUD FOR EARTHQUAKE, FIRE, RIOT

Word clouds have been prepared according to frequency of words. As the font size increases frequency of word used in the given context increases.

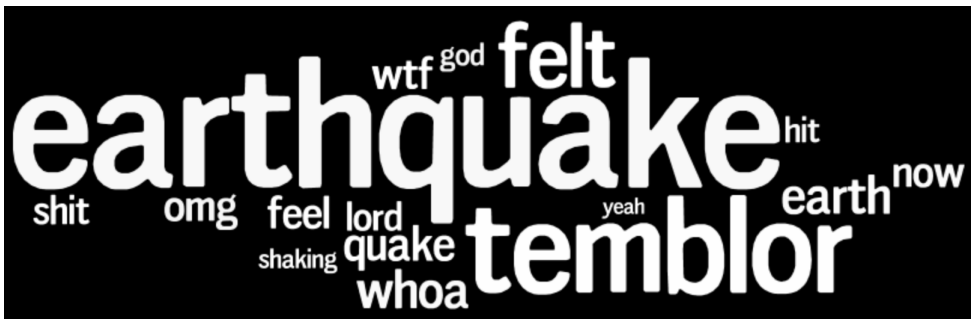


Figure C.1. Frequently used words for earthquake incident.



Figure C.2. Frequently used words for fire incident.

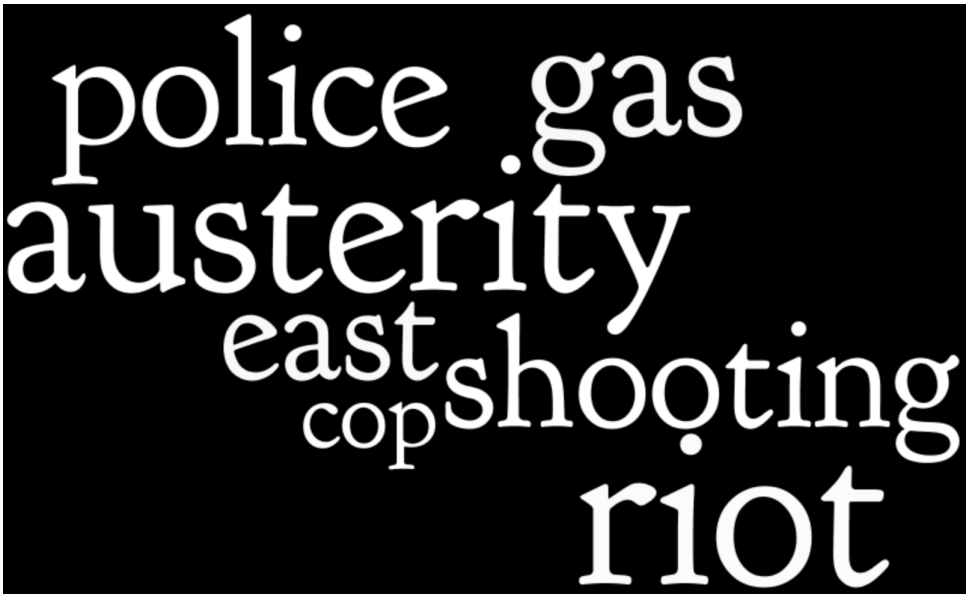


Figure C.3. Frequently used words for riot incident.

APPENDIX D: MICROBLOG POST TEMPLATES

Earthquake and fire incident related user positive, official content contribution templates.

Table D.1. Earthquake incident related user positive contribution templates.

User post
Earth quake [PUNC]
Earth quake
did you feel the quake
Anyone else felt that earthquake [PUNC]
earthquake
did you feel the earthquake [PUNC]
What is that [PUNC] Earthquake [PUNC]
i just felt a [TIME]-second earthquake [PUNC]
Earthquake [EMO]
Did anyone else just feel that [HASH] [PUNC]
Think just felt an earthquake
[MENTION] EARTHQUAKE [PUNC]
Earthquake [PUNC] [PUNC]
earthquake [PUNC]
EarthQuake [PUNC]
earth quack

Table D.2. Earthquake incident related official content contribution templates.

User post
HASH Time: TIME Region:LOCATION Ml: NUMBER Depth: NUMBER km
Earthquake Mag NUMBER TIME LOCATION
Moderate earthquake, LOCATION, TIME, URL
EARTHQUAKE TIME (NUMBER), LOCATION NUMBER NUMBER URL
NUMBER Ml earthquake occurred at TIME, NUMBER km SE of LOCATION, URL
HASH M NUMBER, LOCATION: TIME URL
NUMBER earthquake PUNC TIME near LOCATION URL
HASH HASH M NUMBER,NUMBER km NE of LOCATION, URL

Table D.3. Earthquake incident related negative contribution templates.

User post
feels like an earthquake everytime you come around HASH
I feel like shit ok
I feel like I'm dying
i dont feel like myself
MENTION What it feel like PUNC
Felt like hoopin
I feel really ill
I feel dead
I feel geeky PUNC
Shake some ass PUNC
Feel sick EMOTICON
i can feel your heart beat
Well now I just feel cold
pants on the ground
Earthquake can shake us

Table D.4. Fire incident related user positive contribution templates.

User post
Wicked [LOCATION] picture [PUNC] fire on [LOCATION] [PUNC] [HASH] [URL]
Some kind of fire in the [LOCATION] seen from [LOCATION] [MENTION] [URL]
[LOCATION] is on fire [PUNC] [HASH] [URL]
A shot of the fire in [LOCATION] [PUNC] View from [LOCATION] [URL]
The [ORGANIZATION] that's in fire, on [LOCATION] looks wild [PUNC]
Heading home from [LOCATION] past a fire in [LOCATION]
Just look at the blaze from the bushfires in [LOCATION] seen from [LOCATION] [URL]
[EXAGGERATION] check out this [HASH] pic [LOCATION] [URL]
Fire [PUNC] Looking from [LOCATION] [HASH] [URL]
[HASH] Just took this photo from [LOCATION] [URL]
Fire is travelling east but wind change affecting the area may see the fire travel north east, towars [LOCATION]
Fire on [LOCATION] [PUNC] [URL]
Avoid [LOCATION] at all costs. [LOCATION] blocked off. Big fire [PUNC]
Fire continues to consueme houses on [LOCATION] [URL]
Here is the [HASH] from near [LOCATION] nw of [LOCATION] [URL]
Fire in [LOCATION], know so many people in [LOCATION] [EMO]
I'm on the scene of fire in [LOCATION] [URL]

Table D.5. Fire incident related official content contribution templates.

User post
[LOCATION] / [LOCATION] fire in the walls of a story wood frame dwelling
FIREDEPT [PUNC] Poss Structure Fire reported as black some behind [LOCATION]
[TIME]:[LOCATION], [LOCATION] fire in [LOCATION]
[TIME] [LOCATION]- [HASH] [HASH] [URL]
Fire alarm - water flow, [LOCATION] ([TIME])
Vehicle fire [LOCATION] lanes closed [URL]
[LOCATION] — Vehiclle fire — [LOCATION] — fire in the back lot
Fire Event - [CODE] - [LOCATION] [TIME] [URL]
Check for fire - [CODE] - [LOCATION] - [TIME] - [URL]
Working fire in [LOCATION]

Table D.6. Fire incident related negative contribution templates.

User post
Tryin to get my usher on , but I can't let it burn
Love is friendship set on fire.
Alicia Keys - Girl n Fire
PERSON can die in a fire
MENTION MENTION is the fire on?
LIAR LIAR PANTS ON FIRE!!!!
Your sex is on fire
fire fire fire fire
Love is on fire.
Ice is on fire HASH
Eat healthy, don't smoke and drink, don't do drugs. Die anyway.
URL
Girl and fire
She's On Fire
In just 14 days burn the unwanted fat off your belly URL
I smoke kush
fading away like smoke in the sky
I don't do drugs but I smoke HASH like a champion.

APPENDIX E: MICROBLOG POST SAMPLES

Earthquake, fire, riot, hurricane and flood incident related microblog post samples.

Table E.1. Earthquake incident positive tweets.

Sample tweet
EARTHQUAKE!!!! #JAKARTA
Mandatory post-earthquake tweet
Earthquake!! Just now!!
Awaken by earthquake
Earth quake at 01.57
Wow shoot. An earthquake just now, and I'm on twitter.. My life's a mess
Earth quake, gempa jakarta
Temblor?? #JAKARTA
Earthquake
Temblor en Calama !
#Chinandega fuerte temblor
Yeah right, earthquake. I can see it, from twitter...

Table E.2. Earthquake incident official content tweets.

Sample tweet
Earthquake: M 1.2, Central Alaska. At Mon, 08 Apr 2013 18:45:05 GMT Depth: 22.90 km http://t.co/zjlzfBxDs9
#USGS #alert M 1.5, Central California: January 07, 2013 06:44:14 GMT http://t.co/Z2kp4sFA #earthquake #tsunami #prayfromjapan
Earthquake M 7.5, 94km W of Craig, Alaska: Saturday, January 05, 2013 08:58:19 UTCFriday, January 04, 2013 23:58... http://t.co/iJ3NAus0

Table E.3. Earthquake incident neutral tweets.

Sample tweet
earthquake? or just my feelin? :—
weird....was that an earthquake???
Did an earthquake just happened?
Please tell me that was an earthquake not some supernatural shit
Earthquake is my everyday..

Table E.4. Fire incident positive tweets.

Sample tweet
#chepstowe fire heading straight towards burrumbeet - cfa hoping to stop it before it reaches lake
Grass fire near Little River http://t.co/MgF6RlBd
Little river fire from the train #littleriver #bushfire http://t.co/vrPWTOhX
Heading home from Ballarat past a grass fire in Creswick. Be safe, all. http://t.co/lu6Z5NhI
Fire at Tutuban Mall...
The sky outside my office. Donnybrook Rd grassfire http://t.co/SZAqdW8T
Fast-moving grassfire approaching #Epping North: http://t.co/x1Qel52L
Firefighters battling fire at Green Circle Growers http://t.co/XCfVoR4o
Taxi on Fire at EDSA Kamuning MRT station southbound. http://t.co/oJZ5TOHvB9
Just passed horrible bus fire on 70...bus totally gutted & still burning...appears like everyone evacuated safety thankfully...terrifying!
Other spot fire #soldiercanyonfire http://t.co/0L78uAfXJ4
I 75 east Just Now Car in fire. http://t.co/BCBWG0xz http://t.co/WOxTVdSs http://t.co/4OnLxGxJ
that gas station right where Stellas use to be, on amboy ave, is burning. http://t.co/zLYUaX43
IT'S A FIRE ON 12th ave!!!!

Table E.5. Fire incident official content tweets.

Sample tweet
13:22 Donnybrook Rd, Donnybrook - Grass going (12 appliances, CFA district 14) #bushfire #Fire http://t.co/0W1jZKeF
DONNYBROOK - DONNYBROOK RD , Advice, 18/02/13 1:44 PM - for latest info see http://t.co/2rVz2vW0 #bushfires
Fairfield — Structure Fire — 1589 Rayburn Ct — — 4:47p
Rio Vista 55 — Vehicle Fire — 160 River Rd — fire in the back lot — 3:33p
Check for Fire - E059 - WUTHERING HEIGHTS DR - GLENWYCK ST - 12/29/2012 16:00 - http://t.co/pSy6nCT9
Update: VEHICLE FIRE I-270 SB PAST MANCHESTER RD 2 RIGHT LANES CLOSED (St Louis,MO) http://t.co/B1w93PCw
Fairfield — 5th Alarm Structure Fire — 849 Texas St (Pepperbellys) — 2 story comercian structure / smoke showing — 6:57p
PEPPERELL MA OVEN FIRE 28 HEALD ST

Table E.6. Fire incident neutral tweets.

Sample tweet
This is getting out of hand. #fire http://t.co/RtDG76u
PRAY EMOTICON Stay safe Port Lincoln
Building has split down middle in back. #kare11 http://t.co/8VSZ81UD
Wow the sun is blood red #tasmania #bushfires
The sky over Ballarat has begun to turn red. #ballarat #vicfires #emergency #bushfire #chepstowe http://t.co/cMhjWmLO
What the hell is going on? Fitchburg is burning down.
Fitchburg really is on fire haha
It's raining ash everywhere. I think it's time to head home now. #JurupaFire #fb
A portion of Riverside is in the dark because of the #JurupaFire
Fire time!!!!! http://t.co/8Qx4AFiy
All of Fitchburg is one big cloud of smoke and smells like straight fire right now
Ummm anyone in Carolina forest know what's on fire near my house????? http://t.co/t6irikPRVH
Big as fire smh http://t.co/OfjjKUTv
Mt. Kenya on fire #sunrise #equator #Kenya #Africa http://t.co/w1Kxfhs9

Table E.7. Riot incident positive tweets.

Sample tweet
Crowds are calling for mayor's resignation in Istiklal street near Taksim Square in Istanbul #occupygezipark http://pic.twitter.com/9kWz3jpfOV
It's almost 6 am in Istanbul, the police burn down protesters' tents at #occupygezi http://pic.twitter.com/nX3B2gOepZ via @kiyametprojesi
No blackouts in Taksim so far. Panzers hold positions along Gümüşsuyu and relative calm after Beşiktaş detente established.
Ankara street cloaked in white cloud of tear gas #Turkey pic.twitter.com/inWyWzd7Fq
Boulder County in riot gear on scene http://pic.twitter.com/Krv9tsv2oR
Riot on 13th in boulder holy crap! all I wanted was sushi lol
Riot in Boulder involves 200 people. Assaults, bottle throwing reported. Police blocking 13th St.

Table E.8. Flood, Hurricane incident positive tweets.

Sample tweet
Major flooding in Mesta Park #okwx pic.twitter.com/9DjF2833Zy
Here's a car partially submerged at Council and Reno. @ kfor pic.twitter.com/leFKs36pGI
Car upside down on 39th north of the lake. Emergency vehicles just arrived
Wow, the flooding is real bad in OKC! Dang! Don't go around driving...#OKC #tornado #okwx
Devastating flooding situation getting worse in parts of downtown OKC and surrounding areas. Rain continues @kfor pic.twitter.com/opikY6GjKi
Kentucky ave. 7am. #flashflood #flood #paducah its starting to recede now instagram.com/p/aBCV3TRRIm/
numerous vehicles were trapped in rising waters across Springfield, IL. #ILwx #flood
Flood waters rush in to the Hoboken PATH station through an elevator shaft. #Sandy http://pic.twitter.com/QosgFyOI
Flooding from East River at 20th St and Ave C at Peter Cooper Village #Sandy http://pic.twitter.com/F110yjE8
#Hurricane Sandy is moving between the northeast coast of Cuba and the Central Bahamas.

REFERENCES

1. Ross, B., *How Many People Use the Top Social Media, Apps & Services?*, 2013, `http://tinyurl.com/d3ytpff`, accessed at April 2013.

2. SocialMediadd, *Twitter Stats Infographic and Twitter Facts from 2012 and 2013*, 2013, `http://www.socialmediadd.com/Articles.asp?ID=248`, accessed at April 2013.

3. Bollen, J., H. Mao and X.-J. Zeng, "Twitter mood predicts the stock market", *IEEE Computer*, pp. 91–94, 2010.

4. Bermingham, A. and A. F. Smeaton, "On Using Twitter to Monitor Political Sentiment and Predict Election Results", *Sentiment Analysis where AI meets Psychology (SAAIP) Workshop at the International Joint Conference for Natural Language Processing (IJCNLP)*, 2011.

5. Diakopoulos, N. A. and D. A. Shamma, "Characterizing debate performance via aggregated twitter sentiment", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'10)*, pp. 1195–1198, 2010.

6. Bouillot, F., P. Poncelet, M. Roche, D. Ienco, E. Bigdeli and S. Matwin, "French presidential elections: what are the most efficient measures for tweets?", *Proceedings of the first edition workshop on Politics, elections and data (PLEAD'12)*, pp. 23–30, 2012.

7. Graham, N., M. Yuichiroh, H. Masato and M. Koji, "Safety Information Mining, what can NLP do in a disaster", *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 965–973, 2011.

8. Twitter, *Reporting spam on Twitter*, 2013, `https://support.twitter.com/articles/64986-how-to-report-spam-on-twitter#`, accessed at May

2013.

9. Smith, C., *By The Numbers: 10 Amazing Twitter Stats*, 2013, http://tinyurl.com/mbkv9t6, accessed at May 2013.

10. Ross, B., *Infographic: Twitter Tweet Cheat Sheet To Increase Engagement*, 2013, http://tinyurl.com/kjzu6zh, accessed at May 2013.

11. Twitter, *History of the REST & Search API*, 2013, https://dev.twitter.com/docs/history-rest-search-api, accessed at May 2013.

12. Twitter, *REST API Rate Limiting in v1.1*, 2013, https://dev.twitter.com/docs/rate-limiting/1.1, accessed at May 2013.

13. Twitter, *Using the Twitter Search API*, 2013, https://dev.twitter.com/docs/using-search, accessed at May 2013.

14. Twitter, *The Streaming APIs*, 2013, https://dev.twitter.com/docs/streaming-apis, accessed at May 2013.

15. Twitter, *Error Codes & Responses*, 2013, https://dev.twitter.com/docs/error-codes-responses, accessed at May 2013.

16. Jurafsky, D., *Text Classification and Naive Bayes*, 2010, http://www.stanford.edu/class/cs124/lec/naivebayes.pdf, accessed at May 2013.

17. Christopher D. Manning, P. R. and H. Schütze, *Stemming and Lemmatization*, 2010, http://tinyurl.com/kkfoynp, accessed at May 2013.

18. Gimpel, K., N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan and N. A. Smith, "Part-of-speech tagging for Twitter: annotation, features, and experiments", *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Tech-*

nologies, pp. 42–47, 2011.

19. Ritter, A., S. Clark, Mausam and O. Etzioni, "Named entity recognition in tweets: an experimental study", *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1524–1534, 2011.

20. Li, C., J. Weng, Q. He, Y. Yao, A. Datta, A. Sun and B.-S. Lee, "TwiNER: named entity recognition in targeted twitter stream", *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, pp. 721–730, 2012.

21. Cucerzan, S. and D. Yarowsky, "Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence", *In In Proceedings of the Joint SIGDAT Conference on EMNLP and VLC*, pp. 90–99, 1999.

22. Nadeau, D. and S. Sekine, "A survey of named entity recognition and classification", *Linguisticae Investigationes*, pp. 3–26, 2007.

23. Ikonomakis, M., S. Kotsiantis and V. Tampakas, "Text Classification Using Machine Learning Techniques", *WSEAS Transactions on Computers*, Vol. 4, pp. 966–974, 2005.

24. Daniel de Kok, H. B., *Natural Language Processing for the Working Programmer*, 2013, `http://tinyurl.com/29wsdst`, accessed at May 2013.

25. Lowd, D. and P. Domingos, "Naive Bayes models for probability estimation", *Proceedings of the 22nd international conference on Machine learning (ICML)*, pp. 529–536, 2005.

26. Burges, C. J. C., "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining Knowledge Discovery*, pp. 121–167, 1998.

27. W3, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, 2013,

http://tinyurl.com/lk8ana4, 2004, accessed at May 2013.

28. Mendes, P. N., M. Jakob, A. Garcia-Silva and C. Bizer, "DBpedia Spotlight: Shedding Light on the Web of Documents", *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics)*, 2011.

29. Harrison, G., *10 things you should know about NoSQL databases*, 2010, http://tinyurl.com/7o52g2z, accessed at May 2013.

30. Danilevsky, M., *Uncertain Data and Probabilistic Databases & From SQL to NoSQL*, 2012, http://tinyurl.com/mhnl9hj, accessed at May 2013.

31. Katsov, I., *NoSQL Data Modeling Techniques*, 2012, http://tinyurl.com/7ocdooq, accessed at May 2013.

32. Maxmiliano Franco Braga, F. N. d. L., *Using NoSQL Database to Persist Complex Data Objects*, 2013, http://www.sbpcnet.org.br/livro/63ra/conpeex/mestrado/trabalhos-mestrado/mestrado-maxmiliano-franco.pdf, accessed at May 2013.

33. Aleksandar Milanovic, M. M., *A Survey of Post-relational Data Management and NOSQL movement*, 2013, http://alas.matf.bg.ac.rs/~mi08011/SurveyNoSQL.pdf, accessed at May 2013.

34. DBPedias, *Survey of Distributed Databases*, 2013, http://dbpedias.com/wiki/NoSQL:Survey_of_Distributed_Databases, accessed at May 2013.

35. GrayTelevision, *Firefighters trying to figure out what started fire in Pikeville*, 2013, http://tinyurl.com/m2usb9q, accessed at May 2013.

36. Syracusepost, *Firefighters respond to early morning house fire on the South Side*, 2013, http://tinyurl.com/ms92jpn, accessed at May 2013.

37. AnytownGazette, *Fire Near Tierra Oaks Drive Outside Lake Shasta Contained*,

2013, http://tinyurl.com/lk895je, accessed at May 2013.

38. CranstonPatch, *Firefighters Battle Wellington Ave. Fire*, 2013, http://tinyurl.com/md8r5ps, accessed at May 2013.

39. ABC13, *Fire Damages Bedford County Duplex*, 2013, http://tinyurl.com/m6hesho, accessed at May 2013.

40. KATC-TV, *Fire extinguished at abandoned house*, 2013, http://tinyurl.com/kw8vrdt, accessed at May 2013.

41. KMOV-TV, *Serious fire destroys home in Cahokia*, 2013, http://tinyurl.com/m6xllfb, accessed at May 2013.

42. CBSlocal, *3 Children Injured, 1 Critically, In Southwest Philly House Fire*, 2013, http://tinyurl.com/o6jhvla, accessed at May 2013.

43. Washingtonpost, *Laptop battery causes apartment fire in Manassas, officials say*, 2013, http://t.co/JhIKC9XUKR, accessed at May 2013.

44. WSOC-TV, *Fire tire store, Matthews friday*, 2013, http://tinyurl.com/kpqalk6, accessed at May 2013.

45. Postgazette, *Fire crews responding to fire at giant eagle in Greenfield*, 2013, http://tinyurl.com/msyzov4, accessed at May 2013.

46. BournemouthEcho, *Crews at house fire in Southbourne*, 2013, http://tinyurl.com/khzchue, accessed at May 2013.

47. Huffingtonpost, *Wagner mobile home on fire*, 2013, http://tinyurl.com/lvxzgaw, accessed at May 2013.

48. Northdevongazette, *Chimney fire in Bidegord*, 2013, http://tinyurl.com/kq6w327, accessed at May 2013.

49. Portsmouthgazette, *Fire in shed sends smoke billowing over GoSport*, 2013, `http://tinyurl.com/mz5u2a9`, accessed at May 2013.

50. Petrovic, S., *Real-time Event Detection in Massive Streams*, Ph.D. Thesis, The University of Edinburgh, 2012.

51. Doan, S., B.-K. H. Vo and N. Collier, "An Analysis of Twitter Messages in the 2011 Tohoku Earthquake", *eHealth*, pp. 58–66, 2011.

52. Sakaki, T., M. Okazaki and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors", *Proceedings of the 19th international conference on World wide web*, pp. 851–860, 2010.

53. Mendoza, M., B. Poblete and C. Castillo, "Twitter under crisis: can we trust what we RT?", *Proceedings of the First Workshop on Social Media Analytics (SOMA)*, pp. 71–79, 2010.

54. Qu, Y., P. F. Wu and X. Wang, "Online Community Response to Major Disaster: A Study of Tianya Forum in the 2008 Sichuan Earthquake", *Hawaii International Conference on System Sciences (HICSS)*, pp. 1–11, 2009.

55. Verma, S., S. Vieweg, W. Corvey, L. Palen, J. H. Martin, M. Palmer, A. Schram and K. M. Anderson, "Natural Language Processing to the Rescue? Extracting Situational Awareness Tweets During Mass Emergency", *International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.

56. Zook, M., M. Graham, T. Shelton and S. Gorman, "Volunteered Geographic Information and Crowdsourcing Disaster Relief: A Case Study of the Haitian Earthquake", *World Medical and Health Policy*, 2010.

57. Vieweg, S., A. L. Hughes, K. Starbird and L. Palen, "Microblogging during two natural hazards events: what twitter may contribute to situational awareness", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*

(CHI), pp. 1079–1088, 2010.

58. Palen, L., K. Starbird, S. Vieweg and A. Hughes, "Twitter-based information distribution during the 2009 Red River Valley flood threat", pp. 13–17, 2010.

59. Sreenivasan, N. D., C. S. Lee and D. H.-L. Goh, "Tweet Me Home: Exploring Information Use on Twitter in Crisis Situations", *Online Communities and Social Computing - 4th International Conference*, pp. 120–129, 2011.

60. Cross, A. R., *Social Media in Disasters and Emergencies*, 2010, `http://tinyurl.com/k3d6hbq`, accessed at February 2013.

61. Serdyukov, P., V. Murdock and R. van Zwol, "Placing flickr photos on a map", *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 484–491, 2009.

62. Rattenbury, T., N. Good and M. Naaman, "Towards automatic extraction of event and place semantics from flickr tags", *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 103–110, 2007.

63. Guralnik, V. and J. Srivastava, "Event detection from time series data", *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 33–42, 1999.

64. Sutton, J. N., "Twittering Tennessee: Distributed Networks and Collaboration Following a Technological Disaster", *Proceedings of the 7th International ISCRAM Conference*, 2010.

65. Okazaki, M. and Y. Matsuo, "Semantic twitter: analyzing tweets for real-time event notification", *Proceedings of the 2008/2009 international conference on Social software: recent trends and developments in social software*, pp. 63–74, 2010.

66. Earle, P., D. Bowden and M. Guy, "Twitter earthquake detection: earthquake monitoring in a social world", *Annals of Geophysics*, 2012.

67. Okolloh, O., "Ushahidi or testimony: Web 2.0 tools for crowdsourcing crisis information", *Participatory Learning and Action*, pp. 65–68, 2009.

68. Abel, F., C. Hauff, G.-J. Houben, R. Stronkman and K. Tao, "Twitcident: fighting fire with information from social web streams", *Proceedings of the 21st international conference companion on World Wide Web*, pp. 305–308, 2012.

69. Geert-Jan and Stronkman, *Exploiting Twitter to fulfill information needs during incidents*, M.S. Thesis, Delft University of Technology, 2011.