Assisting Disaster Recovery by Analyzing Mircoblogs

Avideep Mukherjee (18111264), Nayan Das (18111044),
Rahul Sharma (18111270), Soumya Banerjee (18111271)
Instructor : Arnab Bhattacharya
CS685A : Data Mining - Course Project - Group - 08

Abstract

In case of any disaster whether human-made or natural, spreading important information is critical for organizations to handle the situation. Social media becomes an essential tool during disaster times. One such social media tool is Twitter. It provides a vast amount of real-time information. However, extracting useful information from Twitter is challenging due to the high volume of data. Hence, there is a need for developing useful algorithms that could obtain helpful information. In this paper, we are analyzing the tweets that were posted during the 2015 Nepal earthquake. Our goal is to identify tweets that express actionable information like needs and availability of resources which is crucial for effective disaster management.

Keywords: Disaster Management, Information Retrieval, Microblog Retrieval, Classification

1. Introduction

In the present scenario, Social media is an important tool for spreading information during emergencies. Among several social media tools, Twitter is one of the most well know platforms to share data. A good amount of work is done by researchers on Twitter data ranging from Microblog Retrieval (1), (2), Twitter Recommendation (3) and Sentiment dynamics (4), (5).

Information retrieval (IR) from twitter data is usually difficult as most of the tweets are very short and contains a lot of abbreviations and slang language which makes the overall analysis difficult. The real knowledge is hidden beneath a lot of emotional tweets, personal opinions and prayers for the victim.

In this paper, we are classifying tweets during Nepal earthquake 2015 into three classes Need of Resource, Availability of Resource Irrelevant. Each tweet is converted into a numeric feature by employing a Bag of words model. Once each tweet is converted into numeric feature we can use several supervised learning algorithms.

2. Methodology

The overall methodology is divided into 5 steps as follows.

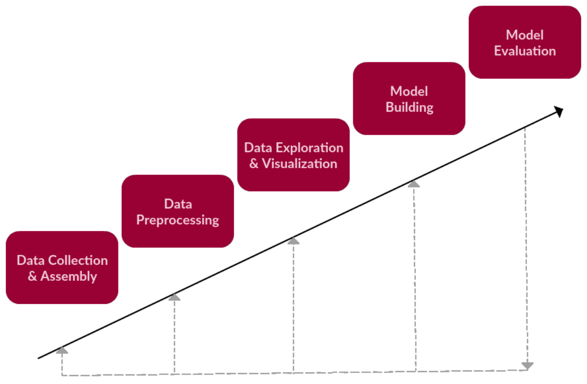


Figure 1: A General framework of the Project[1]

2.1. Data Collection & Assembly

The data-set consists of 70000 tweets that were posted in Nepal earthquake in April 2015. In training data-set, there are 20000 tweets, and test data-set comprises of 50000 tweets. Since public sharing of tweet text is not allowed, a python script was used to access the tweets in JSON format through the Twitter API.

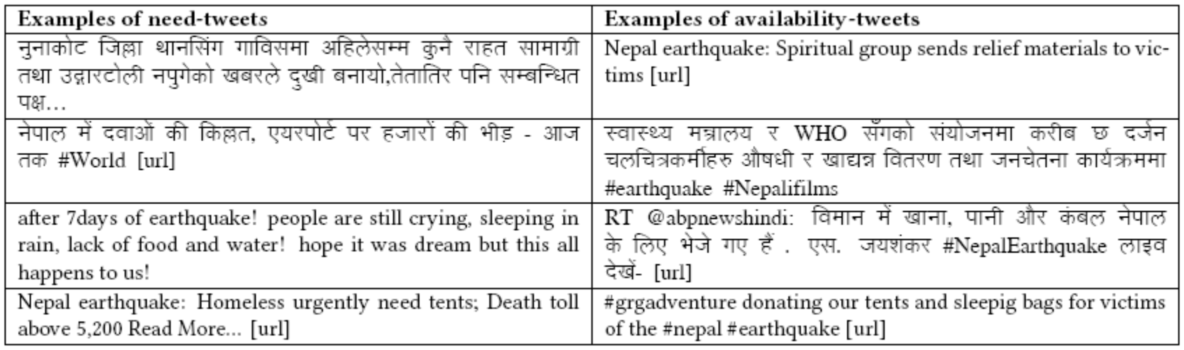
It is not possible to obtain all the tweets as some of the posted tweets are deleted. Hence, the final training data consists of 16932 tweets, and test data-set comprises 40794 tweets. The tweets are in multiple languages like Hindi, English, and Nepali. Most of the tweets are irrelevant and doesn't contain information that could be useful for disaster management. The project aims at developing methodology which could identify *need* tweets and *availability* tweet. The basic definition is as follows

- **Need Tweet**: These tweets inform us about the need or requirement of some resources like food, water, shelter, etc.

- **Availability Tweet**: It indicates the availability of resources like food, shelter, drinking water, Medical facilities, etc.

[1]Source: https://www.kdnuggets.com/wp-content/uploads/text-data-task-framework.png

It is possible for a tweet to be both *need* tweet and *availability* tweet because it indicates the need of some resource as well as the availability of some other resource. The table from (6) explains this concept in more detail.



Examples of need-tweets	Examples of availability-tweets
नुनाकोट जिल्ला थानसिंग गाविसमा अहिलेसम्म कुनै राहत सामाग्री तथा उद्धारटोली नपुगेको खबरले दुखी बनायो,तेतातिर पनि सम्बन्धित पक्ष...	Nepal earthquake: Spiritual group sends relief materials to victims [url]
नेपाल में दवाओं की किल्लत, एयरपोर्ट पर हजारों की भीड़ - आज तक #World [url]	स्वास्थ्य मन्त्रालय र WHO सँगको संयोजनमा करीब छ दर्जन चलचित्रकर्मीहरु औषधी र खाद्यन्न वितरण तथा जनचेतना कार्यक्रममा #earthquake #Nepalifilms
after 7days of earthquake! people are still crying, sleeping in rain, lack of food and water! hope it was dream but this all happens to us!	RT @abpnewshindi: विमान में खाना, पानी और कंबल नेपाल के लिए भेजे गए हैं . एस. जयशंकर #NepalEarthquake लाइव देखें- [url]
Nepal earthquake: Homeless urgently need tents; Death toll above 5,200 Read More... [url]	#grgadventure donating our tents and sleepig bags for victims of the #nepal #earthquake [url]

Table 1: Examples of need-tweets and matching availability-tweets, posted during the 2015 Nepal earthquake

The data-set used in this paper have 4 classes: Irrelevant (C_1), Need (C_2), Available (C_3), Both Need and Available (C_4). As evident from table (1), the data-set suffers from a class imbalance problem.

	N	C_1	C_2	C_3	C_4
Train	16932	16003	211	718	6
Test	40794	39387	427	980	10

Table 1: Data Summary

.

2.2. Data Pre-processing

As discussed in the previous section, the tweets are written in multiple languages. In addition to these problems, there are some other issues which we need to handle. Tweets contain elements like hashtags (#), tagging (@), links, emojis, and pictures which should be removed. There are many inbuilt NLP libraries for English, however, dealing with a language other than English could be difficult. To address all the above issues, we will employ the following strategy.

- Remove elements like hashtags (#), tagging (@) links, emojis, and unicode images from the tweets: These characters are removed from the tweet because they have no meaning. #, @, and link can be removed by following a simple procedure. Let $\gamma \in \{\#, @, \text{http:}\ldots\}$. Whenever we encounter any word of the form γW just delete it.

- Convert all the tweets into English: **googletrans** in python is used to convert all the tweets in English. There are many libraries available for English language, hence it seemed to be a good idea to convert all the tweets into English.

- Remove stop words and employ stemming: Stop words like “a"", “an”, and “the” are removed because they have no meaning. We are using a bag of words model here, so only meaningful words are saved. NLTK library in python is used to remove the stop words. Porter stemmer within NLTK is used for the purpose of stemming. Stemming is done in the text data to convert the word into its base form. For example words “retrieve”, “retrieval”, “retrieving” are changed to “retriev” via porter stemming algorithm.

- Compute Tf-idf score to transform the tweet into a numeric feature: Tf-idf vectorizer within scikit-learn libary is used for converting the tweet into a numeric feature. The tf-idf measures the importance of a word in a tweet. For a word t in document d, the tf-idf score can be computed as

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{1}$$

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D), \tag{2}$$

  where D represents the complete corpus of documents. tf(t,d) (Term frequency) is the number of times a word t occurs in a document d.

- Word vocabulary is also built using tri-gram features(7). Tri-gram features basically tokenizes three-words as a feature. Hence, if the set of training documents contained n words, tri-gram features result in making ${}^{n}C_{3}$ features. Tri-gram features are used so that the feature vector of each tweet carries some semantic information also.

- Use standard ML algorithms on the transformed feature matrix: The algorithms used are explained in detail in section 2.4

2.3. Data Visualization

We have plotted the data cloud for training data-set to visualize it. All the tweets are translated into English. There are no stop words, @,#, and emoticons. In addition to this, porter stemming is done to get a cleaned version of the data-set.

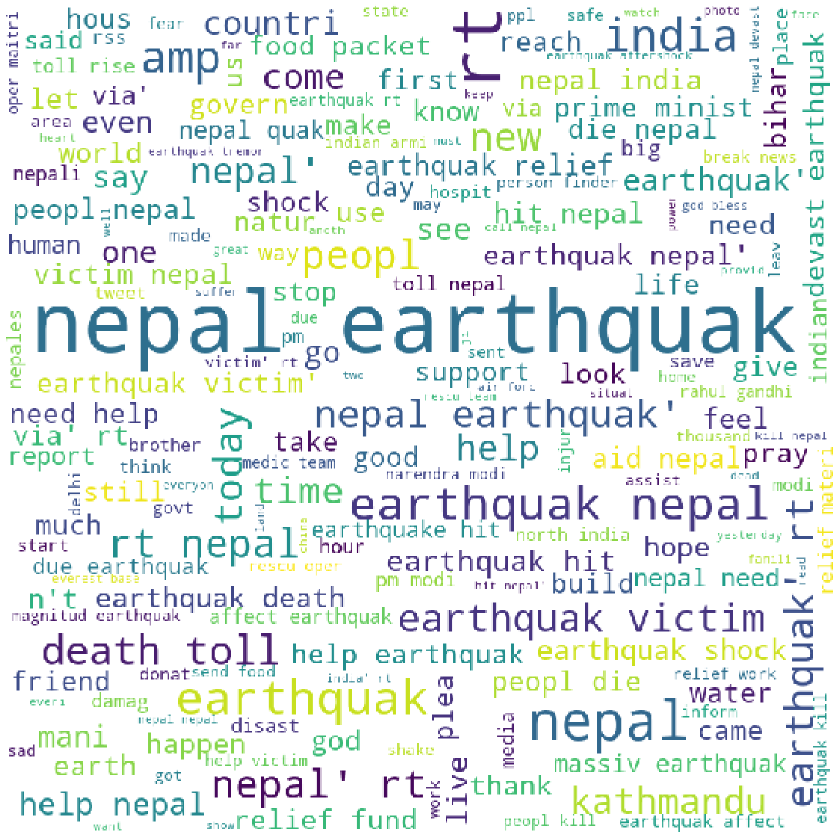


Figure 2: Data Cloud generated after translating the tweets and stemming them

2.4. Model Building

The final datasets that are generated after preprocessing from the raw data are then fit into several classification algorithms. The software and packages used for this task are listed below :

- Python - 3.6
- Scikit-Learn - 0.20.0

The classifiers which are used to classify the tweets are as follows :

- Support Vector Machines (SVM)
- Logistic Regression
- Decision Tree
- Random Forest

Since, the data has quite a bit of class imbalance, therefore slight modification was done during classification. All the classifiers contain an argument of `class_weight` which if set to `balanced` does the following to the training data while building the model.

Weights associated with classes in the form `{class_label:weight}`. If not given, all classes are supposed to have weight one. The `balanced` mode uses the values of `y` to automatically adjust weights inversely proportional to class frequencies in the input data as `number of instances / (number of classes * instances from class y)`.

All the classifiers are well-tuned with different parameters and grid search is performed over all the hyper-parameters with 10-fold cross validation and the best model is taken for classification. The test data are fit into these models and evaluated based on multiple parameters as discussed in section 2.5.

2.5. Model Evaluation

The performances of different classifiers, discussed in section 2.4 are evaluated using evaluation metrics as listed under :

- Accuracy
- Precision
- Recall
- F1-Score

The confusion matrix generated after prediction of each type of datasets on each classifier are also presented so that a fair idea can be derived as the class distribution of the predicted labels. The results for each variation of preprocessing of the data with different classifiers and different evaluation metrices are shown below :

2.5.1. *Support Vector Machines*

Sl. No.	Dataset	`n_gram`	Accuracy	Precision	Recall	F1-Score
1	Total English	1	0.973	0.973	0.973	0.973
2	Total Native	1	0.973	0.973	0.973	0.973
3	Undersampled English	1	0.938	0.938	0.938	0.938
4	Undersampled Native	1	0.943	0.943	0.943	0.943
5	Total English	3	0.972	0.972	0.972	0.972
6	Total Native	3	0.973	0.973	0.973	0.973
7	Undersampled English	3	0.894	0.894	0.894	0.894
8	Undersampled Native	3	0.892	0.892	0.892	0.892

Table 2: Performance of SVM on different variations of the dataset

2.5.2. *Logistic Regression*

Sl. No.	Dataset	`n_gram`	Accuracy	Precision	Recall	F1-Score
1	Total English	1	0.975	0.975	0.975	0.975
2	Total Native	1	0.9749	0.9749	0.9749	0.9749
3	Undersampled English	1	0.933	0.933	0.933	0.933
4	Undersampled Native	1	0.933	0.933	0.933	0.933
5	Total English	3	0.973	0.973	0.973	0.973
6	Total Native	3	0.9749	0.9749	0.9749	0.9749
7	Undersampled English	3	0.903	0.903	0.903	0.903
8	Undersampled Native	3	0.910	0.910	0.910	0.910

Table 3: Performance of Logistic Regression on different variations of the dataset

2.5.3. Decision Tree

Sl. No.	Dataset	`n_gram`	Accuracy	Precision	Recall	F1-Score
1	Total English	1	0.954	0.954	0.954	0.954
2	Total Native	1	0.968	0.968	0.968	0.968
3	Undersampled English	1	0.877	0.877	0.877	0.877
4	Undersampled Native	1	0.849	0.849	0.849	0.849
5	Total English	3	0.958	0.958	0.958	0.958
6	Total Native	3	0.968	0.968	0.968	0.968
7	Undersampled English	3	0.899	0.899	0.899	0.899
8	Undersampled Native	3	0.856	0.856	0.856	0.856

Table 4: Performance of Decision Tree on different variations of the dataset

2.5.4. Random Forest

Sl. No.	Dataset	`n_gram`	Accuracy	Precision	Recall	F1-Score
1	Total English	1	0.9722	0.9722	0.9722	0.9722
2	Total Native	1	0.973	0.973	0.973	0.973
3	Undersampled English	1	0.937	0.937	0.937	0.937
4	Undersampled Native	1	0.942	0.942	0.942	0.942
5	Total English	3	0.972	0.972	0.972	0.972
6	Total Native	3	0.973	0.973	0.973	0.973
7	Undersampled English	3	0.957	0.957	0.957	0.957
8	Undersampled Native	3	0.960	0.960	0.960	0.960

Table 5: Performance of Random Forest on different variations of the dataset

2.5.5. Multi Layer Perceptron

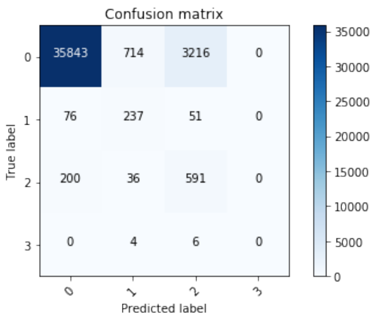
Mutli Layer Perceptron was used only on uni-gram tokenization. Tri-gram wasn't used with MLP due to lack of computational resources and time.

Sl. No.	Dataset	`n_gram`	Accuracy	Precision	Recall	F1-Score
1	Total English	1	0.9725	0.9725	0.9725	0.9725
2	Total Native	1	0.9724	0.9724	0.9724	0.9724
3	Undersampled English	1	0.871	0.871	0.871	0.871
4	Undersampled Native	1	0.869	0.869	0.869	0.869

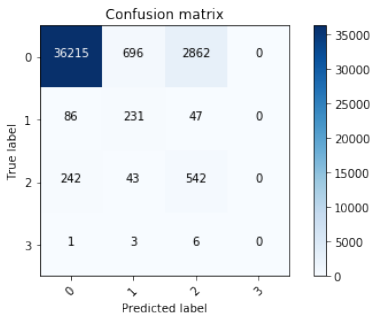
Table 6: Performance of Random Forest on different variations of the dataset

3. Analysis of Results

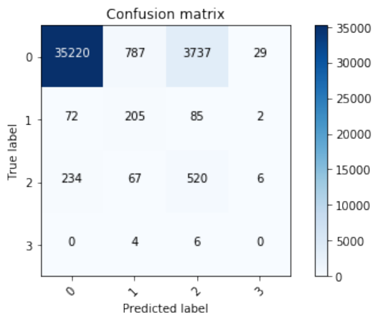
According to the different evaluation metrics, it is clear from the above table that *SVM* and *Logistic Regression* are performing best over the data when the *whole* of data is taken into account and *translated into English*. This implies translating the tweets to English has been an effective step. that However, this might not be the best metric to evaluate the classification task. According to our objective we should be concerned about the amount of false negative rather than amount of false positives. In other words we must weight them differently because a relevant tweet getting classified as a irrelevant one is way more dangerous than the other way round. Therefore we should take into account the confusion matrices produced by every classifier. We are not presenting all the thirty-two confusion matrices, instead below are the best confusion matrices returned by each of the classifiers on a particular configuration of preprocessing. Each caption of each figure is appended with a number n which denotes the nth configuration in it's corresponding table. For example 'SVM Best (7)' means that the 7th configuration of preprocessing in Table 2 has produced the best confusion matrix where the false negative rate is minimum.



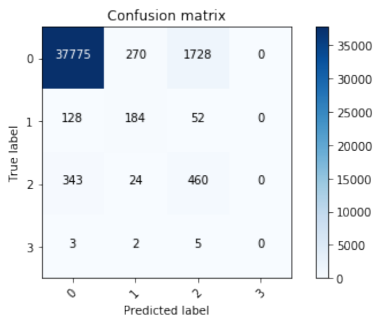
(a) SVM Best (7)



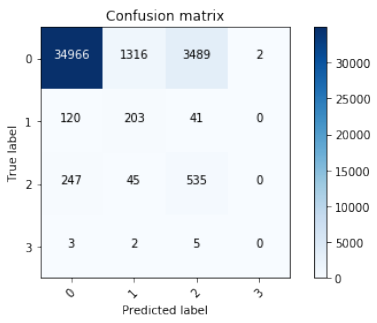
(b) Logistic Best (7)



(c) Decision Tree Best (3)



(d) Random Forest Best (3)



(e) MLP Best (3)

From the five confusion matrices presented above, we can clearly observe that SVM performs best, as in it results in least number of misclassification of relevant tweets as irrelevant. The configuration of preprocessing as mentioned is undersampled and after taking tri-gram features which led to retain some semantic information which in turn helped the classifier to make a better prediction.

4. Conclusion

In this paper, we dealt with a data-set with huge class imbalance. Data cleaning played a substantial role in improving the overall performance of the model. Accuracy in the traditional sense turns out to be a poor indicator of the model performance. False positives (Irrelevant tweets labeled as need tweet or availability tweet) is not that undesirable as compared to False negative (Relevant tweets marked as irrelevant). Hence, the model's performance is judged through accuracy as well as False negative rate. Through experiments, it was found that most algorithms perform well on trigram features with undersampling (SVM, Logistic, and Decision tree). As compared to the raw data which consists of multiple languages, the data which was translated to English yielded better results.

References

[1] Ayan Bandyopadhyay, Kripabandhu Ghosh, Prasenjit Majumder, and Mandar Mitra. Query expansion for microblog retrieval. *International Journal of Web Science*, 1(4):368–380, 2012.

[2] Shangsong Liang and Maarten de Rijke. Burst-aware data fusion for microblog search. *Information Processing & Management*, 51(2):89–113, 2015.

[3] Noor Aldeen Alawad, Aris Anagnostopoulos, Stefano Leonardi, Ida Mele, and Fabrizio Silvestri. Network-aware recommendations of novel tweets. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 913–916. ACM, 2016.

[4] Johan Bollen, Huina Mao, and Alberto Pepe. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *Icwsm*, 11:450–453, 2011.

[5] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[6] Moumita Basu, Saptarshi Ghosh, Kripabandhu Ghosh, and Monojit Choudhury. Overview of the fire 2017 track: Information retrieval from microblogs during disasters (irmidis). *Working notes of FIRE*, pages 8–10, 2017.

[7] Tanel Alumäe and Mikko Kurimo. Efficient estimation of maximum entropy language models with n-gram features: An srilm extension. In *Eleventh annual conference of the international speech communication association*, 2010.

[8] Steven Bird and Edward Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[10] Anastasia Giachanou and Fabio Crestani. Tracking sentiment by time series analysis. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1037–1040. ACM, 2016.

[11] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)*, 47(4):67, 2015.

[12] Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164, 2011.

[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[14] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[15] Stephanie M Strassel, Ann Bies, and Jennifer Tracey. Situational awareness for low resource languages: the lorelei situation frame annotation task. In *SMERP@ ECIR*, pages 32–41, 2017.

[16] Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM, 2010.