Classifying Tweets to Improve Disaster Relief Planning

Tushita Gupta
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania
Email: tushitag@andrew.cmu.edu

Cody Yang
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania
Email: cyyang@andrew.cmu.edu

***Abstract*—Natural disaster response planning can be difficult for first responders. Several problems resulting from the disaster such loss of electricity, restrict the flow of information from victims to responders. With a lack of information, it is hard for first responders to adapt their plans according to the progress of the disaster. However, there is a lot of raw data on social media platforms such as Twitter from which information can be gathered about the extent of a disaster. Information from tweets can help responders form better relief plans. Unfortunately, much of the information is scattered across millions of tweets and it is quite tedious to distinguish useful information about the disaster from everything else on twitter.**

Our goal is to build a system to classify tweets as being relevant or irrelevant to a disaster based on text from the tweets. We will collect tweets from Hurricane Sandy and pre-process the text so that it can be used for machine learning. We will represent the pre-processed text as a vector and we will use the unsupervised machine learning technique, spectral clustering, to represent the data in higher dimensional space. Finally, using the transformed data we will perform k-means clustering to classify the tweets as being relevant or irrelevant. These unsupervised machine learning methods will be the key techniques used in this paper to analyze large datasets of tweets.

Keywords*—*unsupervised learning, spectral clustering, k-means clustering, Twitter, feature vector, tf-idf scoring.

I. Introduction

Disaster relief planning is very complicated due to the ever changing nature of disasters. For this reason, plans created for relief by first responders must be constantly updated with the most recent information about the storm. However, the flow of information is often restricted by disasters temporarily affecting the grid causing power outages, blocking roads with debris etc. Because of these disruptions, first responders aren't able to get to impacted areas and receive information on the status of the affected residents.

The emergence and integration of social media platforms such as Twitter in our lives has given people a new way to share information about events occurring around the world. Twitter is constantly being updated with about 6,000 tweets being posted every second worldwide[1]. There are over 330 million users monthly which means that information about almost every subject is being posted about on Twitter. Thus, during disasters, those affected may post to Twitter about their status and the current condition of the area they live in. With this information, first responders would have the information to develop better relief strategies.

The problem with simply mining all tweets during disasters is that there are also many users posting about completely unrelated topics. Thus, is it becomes necessary for the first responders to read through all the tweets collect and manually filter out the unrelated tweets. This process can be extremely time consuming as there are likely millions of tweets to read through. There is a need for a system to automatically detect what tweets are relevant so the first responders can instantly identify what they need to accomplish in their plans.

To accomplish this automatic labeling of tweets, we can use unsupervised machine learning techniques to cluster the unlabeled tweets into two groups: relevant and irrelevant to the disaster. One popular technique to do this is spectral clustering. Spectral clustering is performed on a similarity matrix to reduce the dimensions of the data and computing the eigenvectors. K-means clustering can then be used on this reduced dimensional data to create k separate groups. The main steps of this process include data extraction and cleaning, feature extraction and unsupervised learning.

II. Literature Review

There have been various studies exploring methods of disaster analysis through social media.

Joe Fan et al[2] attempted to predict Hurricane Sandys impact through a linear regression model and measured the effectiveness of augmenting such a model with social media data by comparing training/testing errors with that of the non-augmented model. However, due to the relatively few number of training examples, the model quickly ran into the problem of overfitting the data. One of the issues with supervised learning methods is obtaining enough labelled data in order to avoid overfitting.

Schulz et al[3] performed sentiment analysis on twitter posts to classify whether or not tweets contribute to situational awareness, such as identifying people in danger during a disaster, but the issue of having a large enough dataset is brought up again. We aim to circumvent this problem through unsupervised learning methods.

Other pieces of literature have examined unsupervised learning techniques of clustering Twitter data based on opinion.

Muqtar Unnisa et al[4] compared unsupervised learning methods and found that spectral clustering gives better results than Naive Bayes, SVM's, and Max Ent. In addition, they concluded that unsupervised learning methods perform better than supervised learning and reduce the need of having labelled datasets.

III. Methods

The proposed system for the automated labeling of tweets is to collect tweets, pre-process them, create feature vectors for each tweet and cluster them using spectral clustering and k-means clustering. We will then check a subset of the tweets labeled by this system with tweets we have manually labeled to determine the accuracy of our classifier.

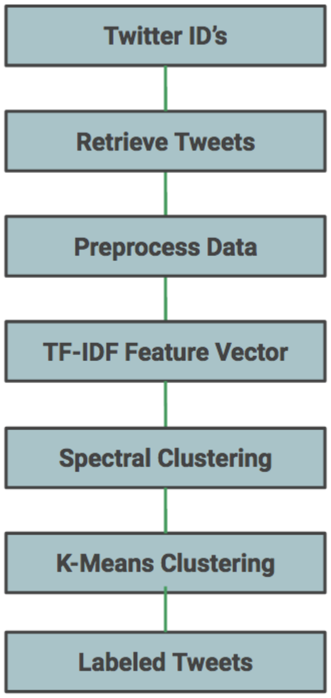


Fig. 1. The proposed system for labeling tweets using spectral clustering.

A. Data Collection

For the purposes of this project we decided to focus on performing unsupervised learning on tweets posted during Hurricane Sandy which occurred in the US between October 22, 2012 to November 2, 2012. We collected 7,042 twitter IDs posted during Hurricane Sandy from an online database, https://goo.gl/Lz9ep8. We then registered an app with Twitter that gave us access to the necessary keys and codes to be able to use the Python Twitter API. Using that, we were able to retrieve the text and date associated with each twitter ID.

B. Data Pre-Processing

One of the biggest parts of the proposed system is to pre-process the data. The raw data from the tweets is filled with noise, unnecessary symbols, unreadable characters, misspelled words, slang words etc. However, to be able to successfully cluster the data, we will need to strip the text of these things in order to normalize it. This pre-processing step will make comparing data from two different tweets significantly easier as the data will be normalized and comparable. Cleaning the data will also make creating feature vectors and performing clustering more efficient. Most of the cleaning process was performed by the Python nltk library, the preprocessor library and regex. The following steps were taken to pre-process the data:

- Tokenization: This process splits text into chunks or tokens based on white-space and punctuation. Each of the tweets are tokenized to split English sentences into individual words. This process would turn the text 'My cat is brown' into the tokens ['my', 'cat', 'is', 'brown'].
- Stop word removal: There is a comprehensive list of stop words in the English language which include functional words such as 'and', 'or', 'the', 'it', 'I', 'me', 'you', 'these' and 'of'. Stop word removes all instances of any of the words from that list from the tokenized text.
- Stemming: This processes reduces each token to its root form by removing prefixes and suffixes. Ex. This process would turn 'processes' into 'process'.
- Emoji, URL and hashtag symbol were removed to normalize the data. Special characters not in the English language were also removed. In addition, all the words were lower-cased.
- Removal of repeated characters: In internet slang, additional characters are often added to words in order to emphasize them. This process removes those extra characters so the word is in its normal English form. For example, 'reallllllly' would turn into 'really'.

C. Feature Extraction

Once the data was segmented into tokens, we wanted to implement a way to maintain some sort of semantic significance within the words. For this reason we decided to use a combination of unigrams and bigrams for feature extraction and selection. Unigrams treat each token separately whereas bigrams treat two consecutive tokens as one item. Bigrams are useful because they add semantic significance. This is shown by the the string "New York". When the string is tokenized the result is 'new' and 'york'. With only unigrams these tokens mean much less than the bigram representation of the string which is 'new york'. For this reason we decided to implement a combination of bigrams and unigrams to extract the features from the text. This means for the string 'New York' the feature words are 'new', 'york' and 'new york'.

D. Feature Vector

We now have a set of feature words or terms for each of the tweets we collected. In order to perform spectral clustering on the tweets we need to represent the tweets in vector space. Representing the tweets or documents as such will allow us to perform the necessary calculations to determine the similarity between two documents and then represent the data in higher dimensional space.

To represent each document we must first create a dictionary of all the terms used in all documents. This dictionary will store the number of documents each term was found in. Then we need to convert each document into a vector representing each term by its tf-idf score. Tf-idf or term frequency-inverse document frequency scoring is one of the most popular term weighting schemes and is used by most digital libraries[5]. The tf-idf score is numerical weight for each term that indicates how important that term is in the corpus. The score for a term increases the more frequently it is in a particular document but also decreases the more frequently the term is seen in the entire corpus. The equations below are used to calculate the tf-idf score for a term.

$$tf(t) = \frac{\textit{\# of times term t appears in a document}}{\textit{total \# of terms in a document}} \tag{1}$$

$$idf(t) = log(\frac{\textit{total \# of documents}}{\textit{\# of documents with term t in it}}) \tag{2}$$

$$score(t) = tf(t) \times idf(t) \tag{3}$$

Using this scoring mechanism we can create a term vector for each document. Let us consider the simplified document space and term dictionary derived from it shown below:

Document Space

document	terms
D1	hi, name, joe
D2	tomorrow, saturday
D3	going, joe, house, tomorrow
D4	best, friend, name, joe

Term Dictionary

term	document frequency
hi	1
name	2
joe	3
tomorrow	2
saturday	1
going	1
house	1
best	1
friend	1

Term Vectors

term	D1	D2	D3	D4
hi	0.201	0	0	0
name	0.100	0	0	0.075
joe	0.042	0	0.031	0.031
tomorrow	0	0.151	0.075	0
saturday	0	0.301	0	0
going	0	0	0.151	0
house	0	0	0.151	0
best	0	0	0	0.151
friend	0	0	0	0.151

In the tables shown above, we have our document space where each document is represent by the extracted feature words. Then we have our our term dictionary which is used to calculate the tf-idf score of each term in a document as showing in the final table.

E. Spectral Clustering

We use spectral clustering to represent the data in higher dimensional space. Spectral clustering is a useful tool for unsupervised learning as labels are not necessary to identify clusters within the data when the data is represented in higher dimensions. To perform clustering, you need a metric of similarity between two data points which in this case is the documents. We want to be able to measure the similarity of every pair of documents that we have. In this way we will construct a similarity matrix where each point is the relative similarity of a pair of documents. If there are N data points the similarity matrix will be an N by N array. Essentially we are creating a graph where each node is a document and the edges between two nodes is the similarity between the documents. If there is no edge between the nodes, then the documents are not similar at all.

We then clusters based on the similarity to points in the group and dissimilarity to points in other groups. To make clustering simpler, we represent the data from the similarity matrix with its first two eigenvectors corresponding with the largest eigenvalues. We use only two eigenvectors to induce the minimized normal cut [6] of the similarity graph where the nodes are the data points and the edges are the similarity score between the two points. Using two eigenvectors will produce exactly two groups as desired. These are the steps of the algorithm[7]:

1) We start constructing the similarity matrix A by using the cosine similarity function. This function measures similarity based on the dot product between document i and document j. More similar documents will have larger similarity values and vice versa for less similar documents according to the cosine similarity function.
$$cos(s_i, s_j) = \frac{s_i \cdot s_j}{||s_i||||s_j||} \tag{4}$$
We then use this value to find the euclidean distance between 2 vectors using the following formula:
$$dist(s_i, s_j) = 2(1 - cos(s_i, s_j)) \tag{5}$$
We use this value to calculate the gaussian similarity value of the two vectors.
$$A_{ij} = exp(||s_i - s_j||^2/2\sigma^2) \textit{ if } i \neq j \textit{ and } A_{ii} = 0 \tag{6}$$
2) Let D be the diagonal matrix where the ith element in the diagonal is the sum of the similarity matrix As ith row. Calculate the Symmetric Normalized Laplacian matrix L. We use this Laplacian matrix because it approximates the minimization of the normalized cut of the similarity graph.
$$L = D^{-1/2}AD^{-1/2} \tag{7}$$

3) To represent the data in higher space find the eigenvectors, v, and associated eigenvalues, λ, of the laplacian matrix L using eigen decomposition as shown below.

$$Lv = \lambda v \tag{8}$$

4) Construct matrix X from the two eigenvectors v1, v2 with the largest eigenvalues such that X = [v1,v2].
5) Normalize each data point in X to be unit length.

$$Y_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^{1/2}} \tag{9}$$

F. K-Means Clustering

K-means clustering is used to partition the Y matrix constructed at the end of spectral clustering into k groups based on a distance function. For this dataset we will set k to two since we want to classify the tweets into two groups which are relevant and irrelevant. We will be using the Euclidean distance between data points from the Y matrix to the centers of the k clusters to recompute new centers. These are the steps of the algorithm[8]:

1) Randomly select k cluster centers or centroids where each point is the number of columns in the Y matrix constructed from spectral clustering. In this case we have two columns since we only kept the first two eigenvectors.
2) Calculate the distance between all centroids and each point in Y. Assign each point to the cluster with the centroid that is closest.
3) Reassign each centroid to be the mean of the points assigned to that cluster.
4) Repeat steps 2 and 3 until no points are reassigned.

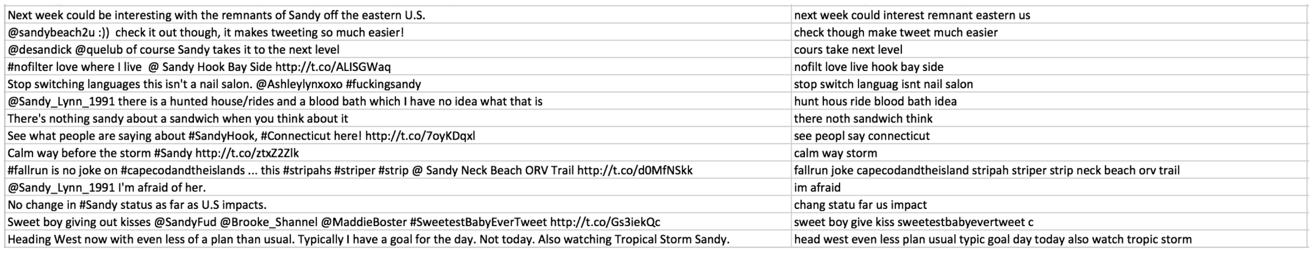
IV. Results

We collected our dataset from an online database containing the twitter IDs of tweets that were posted during hurricane Sandy. From there we cleaned and normalized the data by using techniques such as tokenization, stop word removal and stemming. This results in the text from each tweet being split up into tokens or terms that can be considered features of each tweet. An example of this process is shown in Figure 2.After pre-processing the data we performed feature extraction on the data to only keep relevant information about the data.

Feature extraction was a combination of unigrams and bigrams. A unigram is when each term is treated individually whereas a bigram treats consecutive terms as an individual feature. This helps to maintain semantic significance within the text. After feature extraction, each of the tweets was converted to vector space. This process involved created a dictionary of all the terms in the corpus and counting the number of tweets or documents they appeared in. Each document was then represented as a vector weighted by the td-idf scoring of each term in the document as shown in Figure 3. After each document was vectorized, we created the similarity matrix which contained the similarity score between every pair of documents. The similarity between two documents was defined by the Gaussian similarity function. The similarity matrix was used to create the symmetric normalized Laplacian matrix. The top two eigenvectors were derived from the Laplacian matrix, normalized and clustered via k-means clustering into 2 groups. Values from the top two eigenvectors are shown in Figure 4. Spectral clustering ultimately resulted in 2 groups of irrelevant and relevant tweets as shown in Figure 5. the red cluster had 827 tweets and were classified as relevant. The green cluster had 6217 tweets that were all classified as irrelevant.

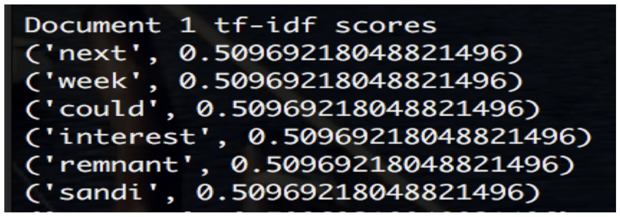
K-means clustering was also performed on the document vectors as a benchmark for spectral clustering. When k-means clustering with k=2 was used on the document vectors, 87 tweets were classified as relevant and 6955 tweets were classified as relevant.

We manually labeled 500 tweets to measure the accuracy of both classifying systems and compare the outcomes. After comparing the labels from spectral clustering to the manually labeled tweets we calculated an accuracy of 84.2%. The accuracy of using k-means clustering on the document vectors alone was 77.6%. This shows us that spectral clustering did improve classification of tweets by 6.6% which is a significant amount. Furthermore, only 11.7% of the tweets were classified as relevant to the disaster by spectral clustering. This means that most of the tweets were irrelevant to the disaster and if a first responder had to manually look through all the tweets to determine relevance, he or she would waste a significant amount of time on irrelevant tweets. Our results show us that spectral clustering yields significantly higher labeling accuracy than k-means clustering alone and that automated labeling can signficantly decrease the amount of tweets responders must look through.



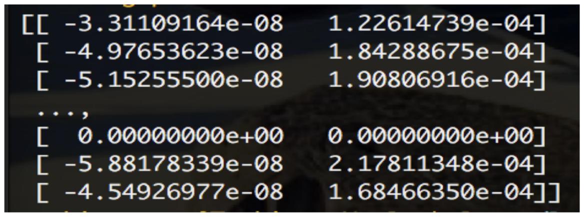
Next week could be interesting with the remnants of Sandy off the eastern U.S.	next week could interest remnant eastern us
@sandybeach2u :)) check it out though, it makes tweeting so much easier!	check though make tweet much easier
@desandick @quelub of course Sandy takes it to the next level	cours take next level
#nofilter love where I live @ Sandy Hook Bay Side http://t.co/ALISGWaq	nofilt love live hook bay side
Stop switching languages this isn't a nail salon. @Ashleylynxoxo #fuckingsandy	stop switch languag isnt nail salon
@Sandy_Lynn_1991 there is a hunted house/rides and a blood bath which I have no idea what that is	hunt hous ride blood bath idea
There's nothing sandy about a sandwich when you think about it	there noth sandwich think
See what people are saying about #SandyHook, #Connecticut here! http://t.co/7oyKDqxl	see peopl say connecticut
Calm way before the storm #Sandy http://t.co/ztxZ2Zlk	calm way storm
#fallrun is no joke on #capecodandtheislands ... this #stripahs #striper #strip @ Sandy Neck Beach ORV Trail http://t.co/d0MfNSkk	fallrun joke capecodandtheisland stripah striper strip neck beach orv trail
@Sandy_Lynn_1991 I'm afraid of her.	im afraid
No change in #Sandy status as far as U.S impacts.	chang statu far us impact
Sweet boy giving out kisses @SandyFud @Brooke_Shannel @MaddieBoster #SweetestBabyEverTweet http://t.co/Gs3iekQc	sweet boy give kiss sweetestbabyevertweet c
Heading West now with even less of a plan than usual. Typically I have a goal for the day. Not today. Also watching Tropical Storm Sandy.	head west even less plan usual typic goal day today also watch tropic storm

Figure 2: Tweets on the left and the extracted terms on the right



```
Document 1 tf-idf scores
('next', 0.50969218048821496)
('week', 0.50969218048821496)
('could', 0.50969218048821496)
('interest', 0.50969218048821496)
('remnant', 0.50969218048821496)
('sandi', 0.50969218048821496)
```

Figure 3: Example tf-idf scores for a document



```
[[ -3.31109164e-08   1.22614739e-04]
 [ -4.97653623e-08   1.84288675e-04]
 [ -5.15255500e-08   1.90806916e-04]
 ...,
 [  0.00000000e+00   0.00000000e+00]
 [ -5.88178339e-08   2.17811348e-04]
 [ -4.54926977e-08   1.68466350e-04]]
```

Figure 4: Values from the first two eigenvectors

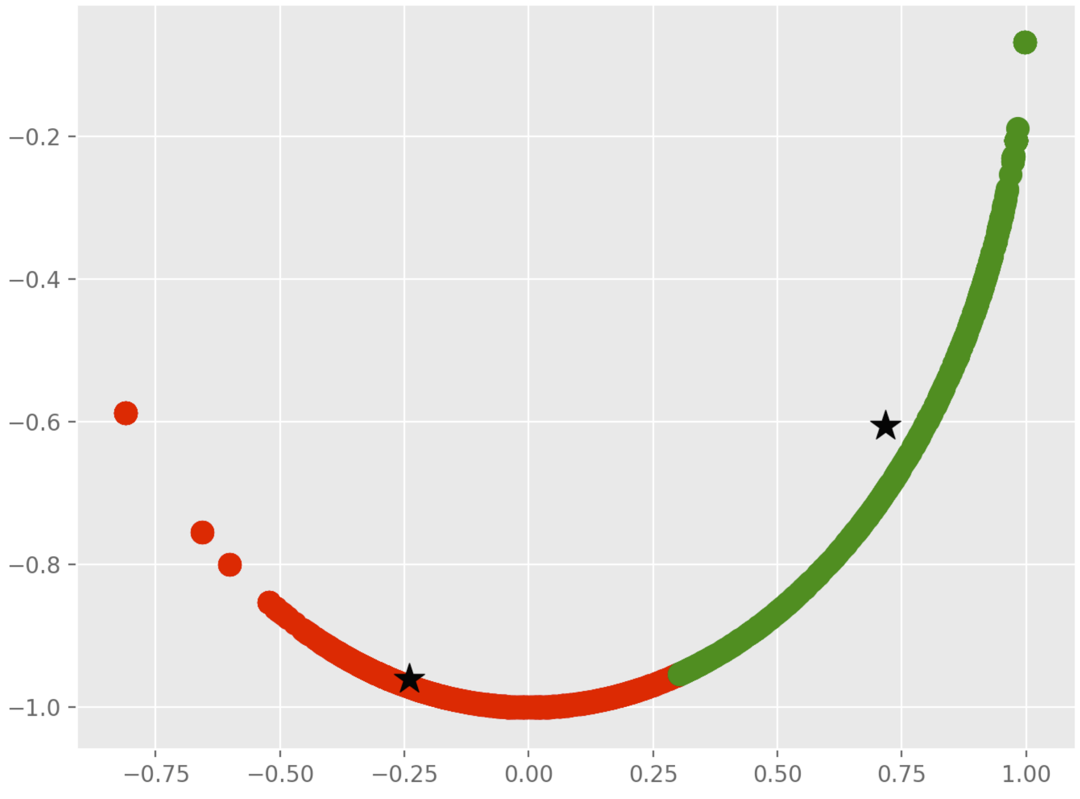


Figure 5: Stars represent centers of the red and green clusters. Green cluster represent the irrelevant tweets and red cluster represents the relevant tweets to the disaster

V. Conclusion

Overall, we present a novel application of spectral clustering in classifying text based on relevance to a crisis with the intention of aiding first responders in planning relief. Furthermore, our results have shown that spectral clustering performs better than K-Means clustering in classifying relevant tweets.

Future work needs to be done on improving preprocessing of the text. We have observed that using unigrams and bigrams to maintain semantic significance improves classifier performance but there is room for improvement in extracting more important features. With some improvements, spectral clustering could transform twitter data into a potentially valuable real-time resource for first responders in a crisis.

References

[1] Krikorian, R. (2013, August 16). Tweets per second. Retrieved December 11, 2017, from https://blog.twitter.com/engineering/en_us/a/2013/new-tweets-per-second-record-and-how.html

[2] Fan, J., Lazrus,, A., Ravulapalli, A., & Ross, M. (n.d.). Intelligent Technology for Innovation in Urban Disasters. Retrieved December 11, 2017.

[3] Schulz, A., Paulheim, H., Thanh, T. D., & Schweizer, I. (n.d.). A Fine-Grained Sentiment Analysis Approach for Detecting Crisis Related Microposts. Retrieved December 11, 2017.

[4] Unnisa, M., Ameen, A., & Raziuddin,, S. (2016). Opinion Mining on Twitter Data using Unsupervised Learning Technique. International Journal of Computer Applications, 148(No.12). Retrieved December 11, 2017.

[5] Breitinger, Corinna; Gipp, Bela; Langer, Stefan (2015-07-26). ”Research-paper recommender systems: a literature survey”. International Journal on Digital Libraries. 17 (4): 305338.

[6] Von Luxburg, U. (2007). A Tutorial on Spectral Clustering. Statistics and Computing, 17(4). Retrieved December 11, 2017.

[7] Ng, A. Y., Jordan, M. I., Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In Advances in neural information processing systems (pp. 849-856).

[8] NK, M. (2017, October 1). K-Means Clustering in Python. Retrieved December 11, 2017, from https://mubaris.com/2017/10/01/kmeans-clustering-in-python/