Twitter Event Detection and Ranking

Honglin Zheng
University of California
Los Angeles
California 90024
hlzheng@g.ucla.edu

Lefan Zhang
University of California
Los Angeles
California 90024
zhanglef7@gmail.com

Weicheng Huang
University of California
Los Angeles
California 90024
wrhuang@ucla.edu

Yao Xie
University of California
Los Angeles
California 90024
allenxie@cs.ucla.edu

Yun Zhang
University of California
Los Angeles
California 90024
zhangmike@g.ucla.edu

ABSTRACT

Nowadays huge amounts of data generated by Social Networking Sites (SNS) probably contains the development of the latest events. In this work, taking Twitter as an example, we aim to classify the tweets into several common event types and cluster tweets with related events so that we can rank the event according to its popularity. We divide the problem into several sub-problems including filtering out noise data, multi-classifying, clustering, ranking and solve each one at each step with trials of multiple methods. Finally we derive a quite reliable and robust model for this problem and analyze the experiment results after training and testing on tweets data we have crawled.

Keywords

Twitter; event detection; data mining

1. INTRODUCTION

The Information Age has provided people with free access to huge amounts of Internet resources. Nowadays, people start to spend more and more time on Social Networking Sites such as Twitter, Facebook, Instagram etc. The popularity of mobile devices boosts this trend and makes those latest posts on these social networks more valuable and frequent. Especially, the past few years have witnessed the boost of monthly active users of Twitter from 30 millions in 2010 to 330 millions in 2017. In this project, we will take advantage of the prevalence of Twitter and utilize the massive amount of data on it to perform in-depth analysis. And with the free access to Tweepy, a Twitter crawling API, it has become much easier to extract useful data pattern from Twitter records.

To get a sense of the four mentioned types of events, we usually refer to official information resources like the local news or the statistics provided by the related organizations such as the Department of Transportation. However, this is usually not real time and requires extra labor to manually collect the information. With the popularity of Twitter, we can develop a model that gathers the intelligence of vast amount of Twitter users to analyze the ongoing events instantaneously.

Given a tweet, we would like to figure out a correct subtype of this event. For example, we will first classify whether it is a relevant tweet message about a traffic event or not. If it is about a traffic event, we will further classify it into a car accident, road work, severe weather condition, or other types of reason. If it is about disaster, we will further classify it into a flood, an earthquake, a volcano eruption, a forest fire and so on. Besides, we want to obtain an accurate summary of this event including keywords in this event, the temporal and geographical information as the output of this data mining model. What is more, we will also try to create a visualization tool to visualize the analysis and prediction results.

The rest of this report is organized as follows. Section 2 will formalize, decribe and give a brief analysis on the problem. The data preparation is introduced in Section 3. In Section 4, this report will introduce the details of implementation for each module. Section 5 will give the experiment results. Related work is provided in Section 6 and our work is concluded in Section 7.

2. PROBLEM DESCRIPTION

Given the tweets data with various data fields, our goal is to find out the related tweets describing the same event and rank them according to their popularity. The problem can be divided into follows.

The first sub-problem is that we need to filter out noisy data from the whole dataset. To simplify the problem, we take those tweets from a certain list of event types as related data while take the rest as noisy data. We need to implement a binary classifier to fulfill this step.

The second sub-problem is that we need to classify those related data into corresponding event types (classes) to a deep granularity so that we can cluster tweets from each event type with a correct and narrowed selection of the whole dataset. This subproblem can be concluded as multiclass classification problem.

The third sub-problem is that we need to cluster those tweets describing the same event into one group, which requires us to implement a reliable clustering algorithm.

The last sub-problem is that we need to rank those events we derived according to the information they contain.

3. DATA PREPARATION

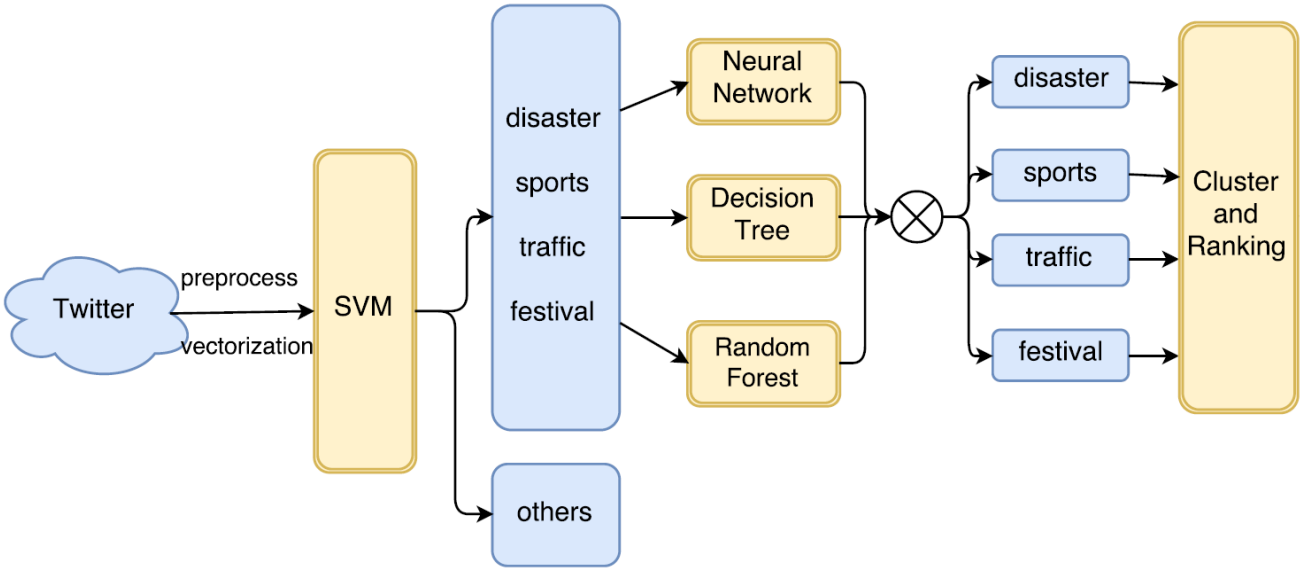


Figure 1: Model Framework

We crawl raw tweets from twitter using twitter API search function. We use four different search queries that are corresponding to traffic, sports, festival and disaster and save the raw tweets in four different json files. Then for each json file, we remove the duplicated tweets and delete fields that are not useful and save them as processed data files. We then label each tweet in the processed data files with labels $0-4$, where 1 is traffic event, 2 is sports event, 3 is festival event, 4 is disaster event and 0 otherwise.

We use the processed data files mentioned above to generate feature vectors. For data file of each category, we extract the text from each tweet, remove stopwords and other meaningless characters, and tokenize the text to be a list of words. We then count how many times each word appears in all the texts and choose the top ten words that appear the most frequently to be the features for this category. This is a feature representation method derived from Bag-of-Words. Since the feature vectors generated by Bag-of-Words usually have high dimensionality and most of the entries are zero, we modify that to count only words which we think it is important to each event type according to the frequency results of the training dataset. Then we generate a feature vector based on these ten features for each tweet text, and each entry in the vector means how many times the feature word appears in the tweet text. Lastly we add an entry to the end of the feature vector, which can be calculated as

$$d = \frac{N_{\text{non\_feature}}}{N_{\text{total}}}$$

where $N_{\text{non\_feature}}$ denote the number of words that are not features and N_{total} denote the total number of words. So the feature vector we get has length 11. Then with the generated feature vectors and the labels, we train an SVM that can classify events under this specific category.

For classification, we simply combine the generated features from the four categories into one, remove duplicates, and then use the combined features to generate feature vectors for all data that has non-zero labels, with every entry in the vector as how many times the feature word appears in the data text. Since we have different features for different categories, the combined feature vector will have a size$\leq$ 40. We will use these feature vectors and labels to train clustering models.

4. MODEL AND METHODS

In order to solve the problem mentioned in this report, we design a model, shown in Figure 1. After preprocessing and vectorizing the twitter data, SVM will be used for filtering out the noise data. Then three methods, Neural Network, Decision Trees and Random Forests are used for classification, followed by DBSCAN for clustering. In the end, we rank the clusters to get the final results.

4.1 Filtering Step: SVM

As we mentioned in Section 4, for data in each category, we generate feature vectors for them based on the category's features. Then we use the feature vectors to train an SVM for this category and this SVM will be responsible of classifying events that belongs to this particular category. It has been proven that SVM is quite efficient in binary classifying and separating tweets according to whether they are positive or negative to a certain event type [3]. We do the same process for all four categories: traffic, sports, festival and disaster and after training we will have four SVMs and each can classify events under a specific category. So when a test data comes in, we will run the four SVMs on the test data, if the test data is classified successfully under any category, it will be sent to the classification step in Section 4.2. Otherwise, the test data will be filtered out as noise.

4.2 Classification Step

Multiclass classification is a frequent problem we encounter in machine learning and data mining process. Given the modified Bag-of-Words feature inputs, we implement two efficient and accurate algorithms to solve it.

4.2.1 Feed Forward Neural Network

INTRODUCTION

Since this is a multiclass classification problem, it is a common method to use the artificial neural network with softmax classifier to solve this. The Artificial Neural Network is based on a collection of connected nodes called artificial neurons and works in a way similar to the human's brain. A signal (value) is transmitted from one neuron to another by multiplying a certain weight of this connection. After each layer, we apply a nonlinear function on the output so that it can identify the potential nonlinear boundary or pattern in the relationship between inputs and outputs. This is the brief structure of an artificial neural network.

METHOD DETAILS

We use PyTorch to build up a feed forward neural network model, a type of the artificial neural network in this work concisely, which is a three-layer neural network structure consisting of an input layer, one hidden layer and one output layer. The dimension of the input layer is 39 according to the feature vector we generated. The dimension of the hidden layer is 200, which is large enough to encode the data pattern in tweets. The dimension of the output layer is 4, which is the number of classes we are going to predict. Besides the structure, the number of training epoch is 5000 and it is enough for the model to reach convergence. The learning rate is set as 0.1 after a number of trials.

When the model reach the hidden layer, the activation (nonlinear) function we use here is ReLU, which replaces values lower than zero with zero. It requires simple mathematical operations and is easy to calculate the derivatives. Our experiment proves that it outperforms sigmoid function. We also apply dropout in our model to prevent the model from overfitting and the dropout rate is set as 0.1. When it comes to the output layer, we normalize the result of those four nodes using log softmax function and pick the class with the largest value as prediction result.

After feeding forward process, we use batch gradient descent (BGD) to update the weights and parameters in the neural network. Since we use ReLU as the activation function, the derivatives for each node are easy to calculate. We repeat those steps mentioned and train the model for 5000 epochs to make sure it has converged. At last given the test feature vector data input, the pre-trained model generates its prediction on the event type (class) each test tweet belongs to. In order to evaluate the performance of the neural network classifier fairly, we use 5-fold validation. Among the 5 trained models, we picked the one with the highest accuracy and apply it on the entire dataset to divide the dataset into 4 different classes. Those classification results are stored to corresponding predict_class_x.vector file and provide the clustering procedure with datasets to continue on.

4.2.2 Decision Trees and Random Forests

In the classification step, we also use Decision Trees and Random Forests as classification methods to compare with the performance of Neural Network.

INTRODUCTION

Decision Trees are a non-parametric supervised learning method that can be used for classification and regression. The goal is to create a model that can predicts the desired value from a set of features, generally a vector of values. In the training procedure, Decision Trees would analyze the relations between the input features and create simple decision rules inferred from them. Then in the predicting procedure, Decision Trees would use the rules learnt above to decide which class the input features would result in or what value could be predicted from the input features.

Nowadays, there are different algorithms designed for Decision Trees to depend on when generating the decision rules. Iterative Dichotomiser 3 (ID3) was developed in 1986 by Ross Quinlan. The categorical feature with the largest information gain is chosen as the current decision rules to divide the current set into several subsets. While in C4.5 which is the successor to ID3, it uses the gain ratio as the decision rules chooser. Classification and Regression Trees (CART) is similar to C4.5, but it constructs binary trees using the feature as well as a threshold that has the largest information gain as a decision rule.

From Decision Trees and the concept of boosting, we come up with using Random Forests to do the classification. Boosting basically says that if one classifier cannot meet the accuracy requirement due to any reason, which is considered weak, we can use multiple classifiers with different weights to achieve the final answer. Adaboost and Realboost are two typical boosting algorithms. Random Forests actually uses the same idea that fits a number of decision tree classifier on various sub-samples of dataset and use averaging to predict. It can improve the predictive accuracy and control over-fitting.

In this project, we use both Decision Trees (an optimized version CART) and Random Forests in the classification procedure [5] [2].

METHOD DETAILS

We use DecisionTreeClassifier and RandomForestClassifier module from scikit-learn to construct our model. For random forest, we specify the number of estimators as 5 and train both of the models using 10-fold cross validation. Among the 10 trained models, we picked the one with the highest accuracy and apply it on the entire dataset to partition the dataset into 4 different classes. We will further perform clustering among these four classes using DBSCAN as below.

4.3 Clustering Step: DBSCAN

4.3.1 Introduction

Clustering is a process of grouping a set of objects which have similar properties into the same group. There are multiple ways to determine whether objects are similar, a major way is to compute the "distance" between two object.

Once we done the classification step, we will get four different dataset each contains a bag of keywords that belong to one type of event. In order to rank the tweets of each type of event, we decided to cluster tweets that in the same group into different cluster, which can be think as another classification step but only classify one type of event each time. This step will run four times because there are four types of event, for each time, the input will be one dataset, and the expected output will be a set of clusters.

4.3.2 Method Implement

We use DBSCAN as our clustering method. The value we are reading in is a csv file, which each line records the

keyword's count and ID of one tweets. And output is a csv file, which each line records the keyword's count, ID and cluster number. There are two initial parameters we have to classify at the beginning of the implementation, which are Eps and MinPts.

- Eps, the maximum radius of the neighborhood, which is the maximum distance we could tolerance to decide whether two tweets are similar.
- MinPts, the minimum number of points in a neighborhood of that point.

In this step, we are using the cosine similarity to measure the distance between two tweets.

$$cos(\theta) = \frac{A \times B}{|A| \times |B|}.$$

We set the Eps to 1, which is only when $cos(\theta) = 1$, the tweets can be assumed to same cluster. And we set the MinPts to 2, which means once there are two tweets that contains same keywords, they will belongs to a new cluster. The clustering process follows this equation,

$$N_{Eps}(q) : \{p \in D | dist(p, q) \leq Eps\}.$$

Object that not contained in any cluster is noise object.

The DBSCAN process start by choosing a random tweet x_i as A, and save x_i to the a list that keep track all visited tweets called L_{visit}, then compare A with all the tweets that are not in L_{visit}, if A has at least MinPts number of neighbors, create a new cluster C, and add A to the cluster, then go over all the neighbors of A, if those tweets have not been visited, add those to the L_{visit}, and doing the same thing as A, check if they have at least MinPts number of neighbors, if so, add them to the neighbor list. If A's neighbors haven't belongs to a cluster, add them to C. After we go through all the tweets, the tweet that are not in a cluster is a noise tweet.

4.4 Ranking Step: Weighted Sum

After the clustering step, we achieve multiple clusters (or say, group) of data. In each group, theoretically, all the tweets are highly related, describing a certain event. In order to predict the most popular event in the future, we need to design a ranking algorithm.

4.4.1 Introduction

For each tweet, we can crawl multiple attributes from Tweepy API. Among them, the following attributes would be useful to the ranking procedure.

- 'retweet_count'
- 'favorite_count'
- 'user_followers_count'
- 'user_friends_count'

Each of the attributes above reflects the popularity of the event or the user in different ways. And they have the following properties:

- 'retweet_count': reflects the popularity of the event. It represents how many people care about this event and how many people would be seeing this.
- 'favorite_count': represents how good the tweet's quality is. It reflects whether people likes the content of the tweet and how likely will they tell others about it.
- 'user_followers_count': reflects how many people would be seeing this tweet
- 'user_friends_count': similar with 'user_followers_count', except that the user might be talking with his/her friends with this content for more times.

According to analysis above, we consider that, for each tweet, we assign different weights based on the four values achieved. The popularity metric for one tweet would be the sum of the four weights, and the popularity metric for one cluster would be the sum of the value of each tweet in the cluster.

4.4.2 Ranking Metric Design

In this section, we show our design of the ranking metric for one tweet and for one cluster. For each tweet, the value of 'retweet_count' and 'favorite_count' would be the main contributor to the ranking. How many users retweet this tweet and how many users favorite this tweet directly reflect the popularity of such tweet. Based on our common observation, the more this tweet is retweeted, the more users will see such tweet and the more popular it will be. Similarly, the more this tweet is favorited, the better or more important the content of this tweet will be, and it will have more chance being recommended to other users. Thus, we consider these two values have a normal contribution to the ranking metric.

The remaining two values, which are 'user_followers_count' and 'user_friends_count', do not contribute as much as the values above, but still have influence on the popularity of one tweet. The value of 'user_followers_count' directly reflects how many users would see this tweet, but keep in mind that not all the followers care about this tweet. That being said, though one user sees a certain tweet, he or she might not talk or share about with others, or even take a closer look at the tweet. But it does have a potential chance that he or she will mention the content of the tweet sometime, someplace. Based on the analysis above, we consider this value has a less contribution to the ranking metric. Same reason as above, we can give similar consideration to the value of 'user_friends_count'.

As a result, we use the original value of 'retweet_count' and 'favorite_count' to measure their contribution to the popularity, and use the log value of 'user_followers_count' and 'user_friends_count'. Thus, for each tweet, let C_{re} denote 'retweet_count', C_{fa} denote 'favorite_count', C_{fo} denote 'user_follower_count', C_{fr} denote 'user_friends_count', and RT denote the ranking metric. In order to avoid the zero value in the log calculation, we plus 1 to each value. Then we can have

$$RT = (C_{re} + 1) + (C_{fa} + 1) + log(C_{fo} + 1) + log(C_{fr} + 1).$$

For one whole cluster, we sum all the ranking metric values to get the ranking metric value for it. However, considering that if the number of tweets in one cluster is too big, it may shadow other clusters even if the four values are not large. In this case, we divide the summation by the number of tweets in one cluster to get the final ranking metric value. Note that this division actually compute the average ranking metric in one cluster. If the event described by one cluster is popular,

Table 1: Accuracy for filtering SVMs

Category	Accuracy	Confusion Matrix
Traffic	0.8864	[[0 6.5] [0 50.9]]
Sports	0.7431	[[4.2 12] [1.3 34.3]]
Festival	0.8808	[[30.3 5.8] [12.5 12.1]]
Disaster	0.6803	[[55.5 3.9] [29.9 16.5]]

Table 2: Accuracy for different classification algorithms

Algorithm	Accuracy	Macro F1 Score
Neural Network	0.845	0.8412
Decision Trees	0.834	0.8385
Random Forests	0.840	0.8394

computing the average will still keep it popular and will not sabotage the final ranking. Under such circumstances, we can have the ranking metric

$$RC_j = \Sigma_{i \in c_j} RT_i / \Sigma_{i \in c_j} 1,$$

where RC_j denote the ranking metric for the jth cluster and c_j denote the jth cluster.

5. EXPERIMENTS

The whole experiment is divided into the corresponding four steps same as the last section.

5.1 Filtering Step

We use accuracy and confusion matrix to evaluate the performance of our SVMs for noise filtering. Since we train four SVMs for different categories, the noise data that will be filtered out are based on the predictions from each SVM. Therefore we will test individual performance of each SVM. In addition, we use 10-fold cross validation and take average of the metrics we measure to determine the performance (Table 1) of our filtering model.

5.2 Classification Step

In this step we evaluate three methods: neural network, decision tree and random forest. In order to demonstrate the performance of those multiclass classification algorithms in all aspects, we utilize three metrics including accuracy, macro F1 score and confusion matrix. Table 2 and confusion matrix visualization graphs will show the performance for each algorithm and the results are all based on 5-fold or 10-fold cross validation.

As you can see, all these three algorithms have achieved similar performance on the whole dataset. Both accuracy and macro F1 score of neural network are a little bit higher than those of Decision Tree and Random Forest. Compared with the result of Decision Tree, Random Forest has a little improvement on performance though the core of the algorithm is to select the best decision tree generated. Besides, the training time of neural network is much longer than the other two algorithms. For our dataset, it takes about 1-2 seconds for Decision Tree and Random Forest to build up and predict while training Neural Network until convergence needs about 2 minutes.

To get a deeper insight into experiment results, we use matploylib to visualize the confusion matrix (Figure 2). The graph of Decision Tree looks quite similar to that of Random Forest so we just compare the results between Neural Network and Random Forest. The Random Forest algorithm has a poor performance on sports data while the Neural Network is not good at identifying disaster data. Both of them are poor at classifying festival data into the correct class.

For Random Forest algorithm, it has the problem of overfitting easily. Since the feature words generated from disaster dataset might also appears in other datasets and the disaster tweets make up a large proportion of whole dataset, it is easy to understand why the result of Random Forest is biased to disaster.

As for Neural Network, it cannot tell the difference between festival tweets and others well. The reason for this is probably festival has a broad scope of concepts and events ranging from film festivals, art festivals, music festivals to sports festivals and so on. The dimension of features selected for festival events might not enough to express the pattern. Without a good representation of features, the neural network could easily get lost during training.

5.3 Clustering Step

After the clustering process, the result will be a file which each line contains keywords count, tweet ID and cluster number.

For example, below are the results for traffic.

- $[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0], 788, 1$
- $[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0], 808, 1$
- $[0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0], 1038, 4$

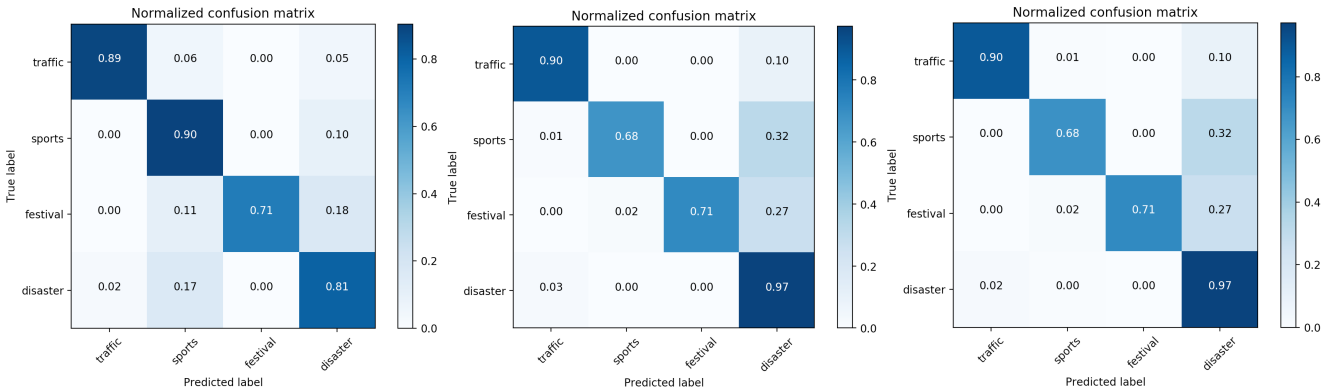
As we can see, the first two tweets with id 788 and 808 belongs to the same cluster, and the third one with id 1038, belongs to a different cluster. We go back to manually check those three tweet texts, below are the original texts.

- 788: '2017 Los Angeles Auto Show Preview, Expect a Giant SUV Traffic Jam: https://t.co/Zp14z9ly51 qua @YouTube'
- 808: '2017 Los Angeles Auto Show Preview: Expect a giant SUV traffic jam https://t.co/iOmQgKxfSJ'
- 1038: 'RT @JoshuaGlotzer: Compensation in Los Angeles Car Accident Lawsuits https://t.co/As7nqVbuTw'

The first two texts talk about the same topic, and the third is unrelated with the first two tweets, thus this clustering is correct.

5.4 Ranking Step

Upon getting the clustered results for the four different classes, we first preprocess it by mapping the tweets' index to the corresponding clusters. Then, we read in the whole set of tweet data, from which we can get the values for 'retweeted_count', 'favorited_count', 'user_followers_count', and 'user_friends_count'. In the end, we traverse all the



(a) Neural Network (b) Decision Trees (c) Random Forests

Figure 2: Confusion Matrix

Table 3: Top 3 results for traffic

1563	I got into a car accident; broke my leg shattered my pelvis; tailbone and my lung collapsed due to airbag collisi...
502	We're already getting reports of heavy traffic in Los Angeles before Game 7. Let's head up to the chopper for a liv...
265	5 years ago today, Kanye West was involved in a near fatal car crash driving home from a recording studio in Los A...

Table 4: Top 3 results for sports

1085	Mike Mitchell didn't hold back when asked about the NFL cracking down on hits.
651	NBA refs ejecting players for anything right now (YOU OUT)
310	What a ride it was. Relive the @astros epic championship run with the 2017 #WorldSeries documentary.

tweets for every cluster in every class, and get the final cluster ranking metric values and the corresponding events described.

For the four classes, the ranking metric values and the corresponding events, shown in the form of the original text, are as follows.

We can see from the results (Table 3) that for traffic, people are most caring about an accident where the victim's lawyer gave up on her. People are retweeting this tweet to help her find a lawyer that could help her. The second result for traffic is about the heavy traffic before a final game for baseball. It also used a video from La La Land to compare to the real situation. The third event is a recall of a rap singer that died in an accident the same day 15 years ago. People have been memorizing him till today.

Table 5: Top 3 results for festival

57	Of today's newspapers Tuba Buyukuston and Angelina Jolie met at the Asia International Film Festival in Los Angeles...
34	David Lynch Eraserhead premiered at the Filmex film festival in Los Angeles, on March 19, 1977. The opening was...
25	il Pump, le rappeur de 17 piges fout le #DaynNight Festival Los Angeles!

Table 6: Top 3 results for disaster

199	RT @ABC7: #earthquake: Preliminary magnitude 4.2 quake strikes near San Diego
124	I think I'm the only person in LA who didn't feel the #earthquake! I don't know if I should be feeling blessed or left out...
79	RT @acornfriend: my favorite thing about the disaster artist is they recalled that ridiculous billboard that was in LA for like 8 years or...

As for the sports results (Table 4), the most popular thing that people are caring about is Mike Mitchell's response towards NFL, which is one of the biggest sports league in the States. The second tweet is a video of imitating the referees in NBA. It is an interesting video and people are enjoying in watching it and sharing it with friends. The third event is a video about the championship in MLB 2017.

The most popular event for festival (Table 5) is the Asia International Film Festival, where Tuba Buyukuston and Angelina Jolie met with each other. People like to see movie stars showing up at the festival and enjoy participate in such global event. The second event about festival is David Lynch's film premiere at the Filmex Film Festival. David Lynch is a great director and would certainly draw lots of eyeballs.

When comes to the disaster (Table 6), people are really caring about the earthquake happened near San Diego. The second one is also about the earthquake, but this time it was in Los Angeles. Recently, there have been several earthquakes happened along the California coast.

5.5 Data Visualization

We also extract the words for each cluster and generate the wordcloud for them to visualize the data. In a wordcloud, the bigger the word, the more popular it is. Below are the

(a) Wordcloud for Traffic

(b) Wordcloud for Sports

(c) Wordcloud for Festival

(d) Wordcloud for Disaster

Figure 3: Data Visualization

four results.

We can see from the figure that, for traffic events, 'car' and 'crash' (Figure 3a) are two main key words. For sports, except 'Los' which is a part of 'Los Angeles', 'Lakers', 'NFL', 'NBA' and 'MLB' (Figure 3b) would be the key words. For festival, 'film' (Figure 3c) festival clearly take the dominating position. For disaster, something happened near 'San' Diego and 'Los' Angeles, referring to the 'Earshquake' (Figure 3d) appears more times.

6. RELATED WORK

Many recent research has focused on the real-time detection of Twitter events by utilizing different information and feature. Carlos's group [3] managed to detect both past and ongoing traffic events utilizing both temporal and geographic information with the help of Named Entity Recognition. They further used real-time clustering to track the event development. Charu's group [1] proposed content-based and network-stream based methods for clustering and event-detection in social streams and showed their advantages over pure text-based methods. Besides, some research work concentrates on taking advantage of Tweets to build up a complete system. Li's group [4] set up a whole framework from crawling valuable data for classifying and ranking to a real-time detecting and analyzing system. Sakaki's group [6] applied semantic analyses and estimated locations to develop an earthquake reporting system.

7. CONCLUSIONS

According to all the analyze we previously give, we come up with a robust process of analysing twitter information and given confidential prediction. During the data preparation stage, we start crawling data using different keywords which could expand the vaiourity of our dataset, we think it is a plus of generating diverse keywords. After we got all the keywords, SVM performed well when we try to filter out the noise datas. And as we can see from the confusion matrix, the output data have at average 75% accuracy which is high enough to let us use those datas as input data for the classification process. We use three algorithms to do the classification, neural network, decision tree and random forest, since we have a quite diverse keywords and a pretty "pure" input data, all three algorithms performed well with at least 83% accuracy and 83 macro F1 score. According to these evaluation results, we believed that all three algorithms are efficient while dealing with key_word_bag type of input data, and with a large scale of training set. The running time lies between 1-2 minutes which is reasonable time duration since the scale of training set is large, and the model we use is quite complicated. In the clustering and ranking stage, even though we only try to use dbscan algorithm and use cosine similarity as the method to compute distance between inputs, without comparison, we could still see they have provided reasonable result, and we believed that cosine similarity is an efficient way to compute the distance when a key_word_bag is given.

As a conclusion, by following all the processes that are described in this report, we could get useful and accurate information from twitter.

Table 7: Task Distribution Form

Task	People
Data crawling and collecting	All
Data preprocessing	Weichen Huang
Implementing SVM filter	Weichen Huang
Implementing Neural Network	Yun Zhang Lefan Zhang
Implementing Decision Tree and Random Forest	Yao Xie Honglin Zheng
Implementing DBSCAN clustering	Lefan Zhang Yun Zhang
Implementing Ranking algorithms	Yao Xie Honglin Zheng
Performing experiments and evaluation	All
Writing the final report	All
Drawing graphs	Yao Xie Honglin Zheng Yun Zhang
Writing the script	Weichen Huang

8. TASK DISTRIBUTION FORM

The task distribution is shown in Table 7.

9. ACKNOWLEDGMENTS

Thank the Intructor, Yizhou Sun, for general project guidance. Thank the Teaching Assistant, Jyun-Yu Jiang, for general implementation suggestions. Thank WordClouds.com for helping with the data visualization. Thank Tweepy API for providing access to crawling twitter data.

10. REFERENCES

[1] C. C. Aggarwal and K. Subbian. Event detection in social streams. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 624–635. SIAM, 2012.

[2] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[3] C. Gutierrez, P. Figuerias, P. Oliveira, R. Costa, and R. Jardim-Goncalves. Twitter mining for traffic events detection. In *Science and Information Conference (SAI), 2015*, pages 371–378. IEEE, 2015.

[4] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang. Tedas: A twitter-based event detection and analysis system. In *Data engineering (icde), 2012 ieee 28th international conference on*, pages 1273–1276. IEEE, 2012.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.