# Happiness Index Detection and Prediction for Twitter Users


| Zhengyuan Liu | Weijie Tang | Di Ma | Xuan Hu | Yuan Tao |
|---|---|---|---|---|
| 604945064 | 305029285 | 004945175 | 505031796 | 305033824 |
| zhengyuanliu@ucla.edu | twjeric@cs.ucla.edu | scarlettmd@g.ucla.edu | x1310317@gmail.com | yuantao@ucla.edu |

Group ID: 11, Group Name: DividedBy0
Email: zhengyuanliu@ucla.edu



## ABSTRACT

Twitter is one of the most widely used social networks in the world. Millions of people express their feeling and propagate hot events on Twitter, providing huge data for data mining and social computing application and research. In this project, we applied data mining and machine learning techniques on twitter data and built an efficient framework to detect and predict twitter users' happiness index. A training corpus of tweets related to significant keywords were built based on twitter data we crawled, and several machine learning algorithms were used to train the classifier of happiness index. We evaluated our classification results and make comparison between different algorithms. Finally, we predicted happiness index for twitter users by their twitter timeline based on our classifiers.



## Keywords

Happiness Index Prediction, Twitter, Data Mining


## 1. INTRODUCTION AND BACKGROUND

### 1.1 Overall Goal

Generally speaking, the main purpose of our project is to detect and predict mood in Twitter by applying data mining techniques on twitter data. Specifically, we aim to detect and predict one dimension of emotion, happiness, rather than multiple dimensions of emotions, and our target is to provide happiness index score for individual twitter user, instead of providing one score for the collective sentiment of a group of people in a special period or area.

### 1.2 Why Twitter

Twitter is one of the most widely used social networks in the world. Millions of people express their feeling and propagate hot events on Twitter every day, providing huge data for data mining and social computing application and research. It is worth mentioning that Twitter has a word limitation. A user can post their twitter with no more than 140 words, so it is convenient for researchers to process tweets and extract keywords.

## 2. RELATED WORK

There has been a large number of pervious work about using data from Twitter or other internet sources to analyze and detect sentiments. A group from University of Texas [1] annotated micro-blog posts in Twitter with seven emotions and studied how these emotions distributed in the data. They also built a classifier that can find sentiments in tweets automatically, which is helpful to develop new emotion prediction techniques related to linguistic style and psycholinguistic knowledge. Saif M. Mohammad [2] successfully tracked and visualized emotions from different types of mail by building a big word-emotion association corpus and comparing these sentiments in different mails. They also discovered that different genders have different tendency to use emotional words in work-place email. Suin Kim [3] proposed a computational frame to study the sentiments in social aspects from Twitter conversations. By applying semi-supervised machine learning method and unsigned data, they observed emotional transitions and emotional influence in Twitter effectively. Johan Bollen and his workmates [4] utilized two mood tracking tools to learn the micro-blogs from Twitter and try to find the relation between emotion states from Twitter and the value of Dow Jones Industrial Average (DJIA). And they found the accuracy of DJIA prediction can be highly influenced by specific public mood dimensions. Also, they performed an emotion analysis [5] of all micro-blogs from Twitter in the 2008 second half. The result show that events in the social, political, cultural and economic sphere did an immediate and highly effect on the various dimensions of public feelings.

## 3. PROBLEM DEFINITION AND FORMALIZATION

The object of our project is to detect and predict happiness index for twitter users according to their tweets. In order to simplify the problem, we suppose that a tweet represents its owner's mood at its created time. Therefore our mood detection and prediction project can be divided into three major subtasks. First, we need to build a model to map features of one tweet from an individual user to a numeric happiness index, which can be easily formalized into a data mining/machine learning problem. Second, we use the model to predict happiness index for all twitter users in our corpus based on their tweets from timeline and get a curve of happiness index for each user. The third subtask is data crawling based on our framework, which will be described in the next part in detail.

## 4. DATA PREPARATION

### 4.1 Twitter Crawling Strategy

We use Tweepy [6-7], a python package, to crawl tweets. Considering the massive amount of twitter data, it is more realistic for us to only crawl tweets that tend to reflect users' emotion. In order to achieve that, 10 keywords are chosen to filter tweets, because we believe they are more likely to evoke emotion on Twitter. As shown in Table 1, the corresponding 5 happy keywords and 5 sad keywords are used to filter tweets. In addition, we focus on the tweets in Los Angeles by constraining the location of tweets while crawling.

**Table 1: Keywords for Filtering Tweets**

| Keywords | Label |
|---|---|
| "Christmas" | Happy |
| "Thanksgiving" | Happy |
| "New Year" | Happy |
| "Halloween" | Happy |
| "Valentine" | Happy |
| "mass shooting" | Sad |
| "Trump" | Sad |
| "Tax Reform" | Sad |
| "Terrorist attack" | Sad |
| "Hurricane" | Sad |

We implement two different ways by Tweepy to crawl twitter data. The first method utilizes Cursors objects and search function to get tweets. The search function would return tweets containing certain keywords within an area, and using Cursors objects enable us to get and handle large number of tweets at once. However due to the rate limit of Twitter API, the crawler is programmed to sleep for 15 minutes before a new search function is called. Another method uses a stream listener to directly listen to real-time tweets, and returns filtered tweets according to the keywords and the location. In this way, the crawler can keep listening without any sleep, thus increases the speed of the crawler. However, because the listener is only listening to the real-time data, it won't have any information regarding retweets, favorites and other important feature. So in the end we have to go with the first method since it retains more versatile feature information (streaming_datafetch_1114.py). We divided the task into 5 parts and each member crawled 2 keywords. The final training dataset scale is around 4000 after data clarification (removing retweets and duplicated texts).

And for the testing dataset, we found 3 representative accounts: Disney, Donald Trump and God (which is an account complaining and making fun of the social figures). These 3 accounts are representatives of Happy, Neutral and Negative accounts that we picked. We can use user id to search for a specific user's timeline and therefore obtain all the tweets belong to his record (streaming_data_userid.py). Each account offers around 800 data for testing purpose. The basic information of these 3 representative accounts is shown in Table 2

**Table 2: basic information of 3 representative accounts**

| Full Name | User ID | Followers | Total Statuses | Screen Name |
|---|---|---|---|---|
| Disney | 67418441 | 5,758,836 | 9,557 | Disney |
| Donald J. Trump | 25073877 | 43,748,882 | 36,517 | realDonaldTrump |
| God | 655093 | 248,435 | 1,392 | god |

## 4.2 Tweets Data Structure

The StreamListener will return the corresponding Status object with the limit of location, keyword and language. The basic data structure of the Status class can be found at https://gist.github.com-/devtechmoe/ef676cdd03ac47-ac503e856282077bf2 [8].

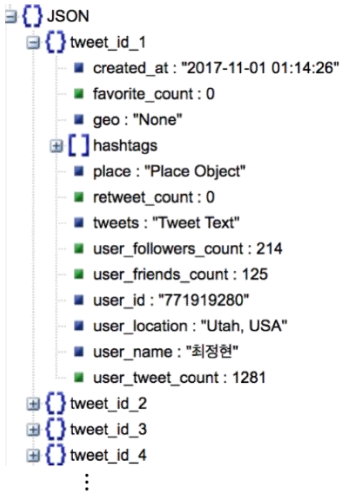


**Figure 1: Crawled Tweets Data Structure**

The crawled tweets are formed in the similar data structure and are saved in separate json files according to the event they belong to. The structure of crawled tweets is shown in figure 1. Each json file contains a list of tweets with unique tweet id. For each tweet, the useful information about the tweet itself and the user who post the tweet is extracted, such as the text and the creation time of the tweet and the id, name, friends count and tweets count of the user.

## 4.3 Data Source Other Than Twitter

In the later training process, we have adopted two more datasets for the training purpose of NLP part. We first used the Twitter Sentiment Analysis Dataset which contains 1,578,627 classified tweets, each row is marked as 1 for positive sentiment and 0 for negative sentiment. This dataset is used to train the nlp model in order to build a more complete dictionary. Due to the large size of the dataset (over 120MB), we split the datasets into 4 parts and only use ¼ of it. We also used the labeled movie review sentiment dataset from Stanford NLP lab. This dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser and was labeled with label 1-5 representing negative to positive. This dataset is majorly used to train the NLP model due to its accuracy and proper size.

# 5. METHODS DESCRIPTION

## 5.1 Overview

Our method has two major steps as discussed in Part 3 Firstly, we build a model to map one tweet from an individual user to a numeric happiness index we define. Secondly, we use the model to predict happiness index for twitter users in our corpus based on their tweets from timeline and get a curve of happiness index for each user. The detailed steps of our method is described below.

## 5.2 Mapping Tweets to the Happiness Index

Obviously, this step can be formalized into a data mining/machine learning problem. The general idea is to build a corpus of tweets with manually-annotated happiness index as our training data; then we define and extract features of tweets and use tweets in the corpus and their features to train a classifier; finally we use the classifier on test data (all tweets from every individual in our dataset) in order to get sequential data of happiness index for all users.

### *5.2.1 Definition of Happiness Index*

First of all, we define a happiness index of a tweet to describe the happiness status for its owner. In order to simplify the problem and reduce the workload of manual annotation, we simply define a both descriptive and numeric happiness index for a tweet with 5 levels: Happy (2), Positive (1), Neutral (0), Negative (-1) and Sad/Angry/Hate (-2).

### *5.2.2 Construction of the Corpus*

Second, we build a corpus of tweets as training dataset which contains enough number of tweets with their corresponding manually-labeled happiness index. Based on our experience, tweets that express happy or unhappy emotion is rather fewer than non-emotional tweets. So if we just crawled tweets without selection, Neutral (0) tends to be the very majority of happiness index in our corpus, which may lead to a bias when using machine learning methods to train the dataset. Therefore, we need to reduce the number of non-emotional tweets in the corpus and make a relatively balanced distribution of our chosen happy index in our training dataset.

**Table 3: Basic Statistics of the Corpus**

| Keywords | Crawled tweets | Labeled tweets | Happiness Index Distribution | | | | | Category |
|---|---|---|---|---|---|---|---|---|
| | | | -2 | -1 | 0 | 1 | 2 | |
| Christmas | 2238 | 729 | 8 | 42 | 445 | 193 | 41 | Happy |
| Thanksgiving | 3153 | 0 | 0 | 0 | 0 | 0 | 0 | Happy |
| Halloween | 594 | 129 | 4 | 27 | 40 | 38 | 20 | Happy |
| birthday | 1698 | 637 | 1 | 21 | 50 | 330 | 234 | Happy |
| Valentine | 253 | 269 | 8 | 16 | 164 | 59 | 22 | Happy |
| mass shooting | 1275 | 502 | 110 | 253 | 133 | 5 | 1 | Sad |
| Trump | 1251 | 510 | 55 | 150 | 242 | 56 | 7 | Sad |
| tax reform | 1125 | 490 | 37 | 108 | 328 | 14 | 3 | Sad |
| Terrorist | 724 | 310 | 191 | 66 | 48 | 4 | 1 | Sad |
| Hurricane | 310 | 191 | 34 | 34 | 74 | 44 | 5 | Sad |
| Sum | 12621 | 3767 | 448 | 717 | 1524 | 743 | 334 | - |

The idea is, to crawl tweets related to several keywords, in which happy or unhappy emotions will likely be expressed related to these keywords. Therefore, we chose 10 popular keywords in recent time and we believed that they are very likely to evoke emotion on Twitter, where 5 of them are happy keywords while another 5 are unhappy keywords. These keywords are shown in Table 1.

Then we crawled twitter data based on the 10 keywords above, and manually label the happiness index for the tweets in we crawled to build the corpus. Because of the large volume of the corpus and limitation of time, we only labeled part of the tweets in our corpus. Basic statistics of our corpus is shown in Table 3.

From Table 3 we can see that for the happy keywords, tweets with happiness index 1 and 2 relatively outnumber tweets with happiness index -1 and -2, vice versa. Therefore data unbalanced problem is kind of relieved and our data crawling strategy is reasonable. However, this problem still existed.

#### *5.2.3 Features Extraction and Selection*

After the construction of our corpus, the key problem here is how to choose effective features of tweets which are likely to reflect or affect people's happiness. Intuitively, the first choice is the text of a tweet. Therefore, we decided to use number of question mark (?), number of exclamation mark (!) and ratio of uppercase of the tweet text as features. Based on NLP techniques, we also got a mood score for each tweet (More details in 5.2.4). Moreover, created time, number of favorite and number of retweet of a tweet are also used as features. Finally, we applied some attributes of tweets' owner as features, i.e. number of followers, number of friends, and number of tweets. In order to verify the effectiveness of our features, we used linear regression to do a selection of our features.

#### *5.2.4 Feature Based on Natural Language Processing Techniques*

To make better use of the tweets which we believe should be a vital part of the mood detection, we obtained 2 large labeled text sentiment datasets and use them to train a NLP model to produce more accurate NLP prediction data. We first preprocessed the input text and then tokenized the text into tokens. Then we created a lookup dictionary of all the unique words and considering the limitation, we set the limit of the dictionary to only take the most frequent 10000 words. Then it was followed by building the One-hot matrix for each recorded word so that each string can be represented with a set of digits and be processed to the layers in neural network. Next we constructed the model with Keras library. To maintain a balance between accuracy and performance, we added 2 dense layers and 2 dropout layers with different activation functions (relu, sigmoid and softmax) to avoid overfitting. The final dense layer would categorize the prediction into 5 categories. We can achieve accuracy around 70% when only distinguishing mood between positive to negative. But matching the exact score would reduce the accuracy to around 25%.

#### *5.2.5 Training of Classifiers*

Based on the features we extracted, we trained a series of multiclass classifiers based on several machine learning algorithms, i.e. linear regression, logistic regression, support vector machine (SVM), neural network and XGBoost [12], and used the Cross-validation (k-fold) method to adjust the parameters and evaluate our classifier's accuracy.

### 5.3 Happiness Index Prediction

We chose the classifier with parameters achieving the best performance as our final model, and used this model to predict happiness index for twitter users in our testing dataset, i.e. tweets from 3 representative accounts. We used a line chart of happiness index for each user to visualize our results.

## 6. EXPERIMENTS

We used linear regression, logistic regression, SVM, neural network and XGBoost [12] as classification algorithms and evaluated these algorithms by running Cross-validation (k-fold) method on our training corpus. We used Python to implement our algorithms. For result evaluation, we used accuracy as our major evaluation method because our task a multiclass classification task. Considering that it is relatively inessential if the classifier miss classifies 1 with 2 or -1 with -2 because they both means positive or negative, we define a weak accuracy to evaluate the accuracy of the classification of positive (1 and 2), neutral and negative (-1 and -2).

### 6.1 Training Dataset and Data Preprocessing

Basic statistics of our training dataset is shown in Table 4.

**Table 4: Training Dataset**

| Number of Tweets | Happiness Index Distribution | | | | |
|---|---|---|---|---|---|
| | -2 | -1 | 0 | 1 | 2 |
| 3496 | 441 | 679 | 1396 | 682 | 298 |

By our data crawling strategy, the unbalanced data problem is relieved but still exists. Therefore, we used random oversampling method to solve it, i.e. re-sampling tweets randomly to make all classes have the same number of tweets. The python package imbalanced-learn [13] is used to do the oversampling. Because of oversampling, if we first do the oversampling and then use k-fold,

the same training sample could appear in both training set and test set, which result in overfitting. In order to avoid this problem, we first do the k-fold and then conduct oversampling on the training data set.

Also, to avoid a feature being dominant due to its relative large range of value, we use the StandardScaler of scikit-learn [14] to apply Z-score normalization on the features.

## 6.2 Classifiers and Result Evaluation

### *6.2.1 Linear Regression and Feature Selection*

From our dataset, we cleaned and preprocessed the data to get 10 candidate features:

- "*user_friends_count*": The number of users this account is following;
- "*user_followers_count*": The number of followers this account currently has;
- "*user_tweet_count*": The number of tweets issued by the user;
- "*retweet_count*": The number of times this tweet has been retweeted;
- "*favorite_count*": The number of tweets this user has liked in the account's lifetime;
- "*exclamation_number*": The number of exclamation marks in this tweet;
- "*length*": The length of this tweet;
- "*question_number*": The number of question marks in this tweet;
- "*uppercase_ratio*": The ratio of uppercase letters in this tweet;
- "*nlppred*": The mood score of this tweet, got by NLP technique.

To decide which features would be useful, we used the scikit-learn [14] to conduct linear regression. The mean squared error (MSE) after 10-fold cross-validation process is used to analyze the performance of the model. The MSE of the model using all 10 candidate features is around 1.82. Then we drop each feature to get the corresponding new MSE value. We only select features that lead to a higher MSE if they are missing. In the end, the average MSE value is around 1.81, and the features we selected for other models are: "*user_friends_count*", "*user_followers_count*", "*retweet_count*", "*exclamation_number*", "*length*", "*question_number*", "*uppercase_ratio*", "*nlppred*".

### *6.2.2 Logistic Regression*

We also used the scikit-learn [14] to conduct logistic regression. The accuracy after 10-fold cross-validation process is used to analyze the performance of the model. The final average accuracy of the logistic regression model is 27.13%, and the weak accuracy is 51.19%.

### *6.2.3 Support Vector Machine*

We used the LIBSVM [15] to realize our SVM classifier. The type of SVM is C-SVC (C = 4), the kernel type is radial basis function. 10-fold cross-validation method is used to adjust the parameters and evaluate the classifier's accuracy. The final average accuracy of the SVM classifier is 42.91%, and the average weak accuracy of the model is 56.32%

### *6.2.4 Neural Network*

We used Neural Network to build a classifier and identify the type of new data. The learning rate is 0.1 and we randomly set the initial weights from(-1,1). Here we used 10-fold cross-validation to calculate the average accuracy of Neural Network and updated the weights each time after loop. We randomly choose 1 out of 15 data from the training data set to train the model. The final accuracy is around 35% based on -2 to 2 distribution. If we divided mood into three categories as negative, neutral and positive, the average accuracy is around 53%.

### *6.2.5 XGBoost*

We used XGBoost [12], an "Extreme Gradient Boosting" method, to do the multi-class classification too. First use grid search with cross-validation in sklearn to find the best parameters of the XGBClassifier and save the best model for prediction tasks. The main parameters are: *eta*=0.3, *max_depth*=6, *min_child_weight*=11, *subsample*=0.8, *colsample_bytree*=0.7, *num_round*=150. Then use those parameters to train the model with hold-out validation and cross-validation and do the classification. The final test error is around 50.7%, which means the accuracy is around 49.3%. The weak accuracy is around 66.7%.

## 6.3 Comparison of Classification Algorithms

The results of all 5 classification algorithms we used are shown in Table 5. Linear classification is what we try first since it is easier to use and interpret. So the first two model we tried are linear regression and logistic regression. The results show that the mean accuracy of the two linear classifications are rather low and it's not hard to understand because the linear model is too simple to classify the tweets. Given that it's hard to get a good fit with linear classification, we tried nonlinear regression models: SVM, Neural Network, and XGBoost. The result of the three nonlinear classification algorithms are better, especially for SVM and XGBoost. The accuracy of Neural Network algorithm is relatively lower since we only constructed one hidden layer. From the line charts predict by SVM and XGBoost (details in 6.4), we can see that although the fluctuation ranges of the line charts generated by SVM are relatively larger than XGBoost, the trends of each figure predicted by SVM and XGBoost are roughly the same.

**Table 5: Comparison of Classification Algorithms**

| Algorithms | Accuracy/MSE | Weak Accuracy |
|---|---|---|
| linear regression | 1.81 | - |
| logistic regression | 27.13% | 51.19% |
| SVM | 42.91% | 56.32% |
| neural network | 35% | 53% |
| XGBoost | 49.3% | 66.7% |

## 6.4 Result Visualization

We used SVM and XGBoost as our final classifier since these two methods achieved best accuracy, and run them on the testing dataset and get line charts for the 3 representative accounts: Disney (Figure 2), Donald Trump (Figure 3) and God (Figure 4). We used plotly [16] to visualize the results. Each line chart shows the the time flow of user's happiness index. To improve readability, when doing visualization, by default we aggregated the happiness index by date. Actually, every visualization result is an interactive graph in an HTML file. One can explore the graph by zooming in and out, dragging, and hovering over the data points.


**Figure 2: Happiness Index (log) Line Chart predicted by (a) SVM and (b) XGBoost for Disney**


**Figure 3: Happiness Index Line Chart predicted (a) SVM and (b) XGBoost for Trump**

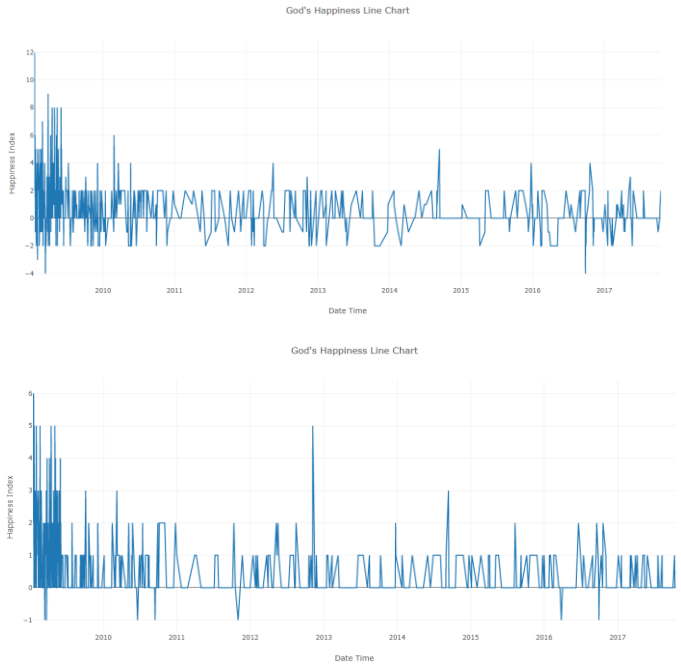


**Figure 4: Happiness Index Line Chart predicted (a) SVM and(b) XGBoost for God**

# 7. DISCUSSION

Accuracy of our happiness index classification is not very high. We think there are four main reasons which lead to a relatively low accuracy.

The major problem is the quality of our training corpus. Because we do not have enough time to manually label all the crawled data, our corpus is not big enough. Moreover, our training data may not be representative enough. For instance, there are a lot of advertisements in the tweets crawled by keyword 'Halloween'.

The second reason is that twitter users express their meaning and mood on Twitter in a way that different from daily life and written text, which is more casual and contains a lot of Internet languages. Moreover our group members are not native English speaker, so we have difficulty labeling some tweets. Moreover, about 60% of the crawled tweets are short sentences (< 30 words) which cannot provide enough context information for the neural network to get the NLP score.

The third reason is that multi-classification is harder to get better accuracy than binary classification. Use the NLP analysis as an example, classifying between positive and negative has an accuracy around 70%. But matching the exact label (-2, -1, 0, 1, 2) would significantly reduce that accuracy. That's why we use a weak accuracy as our evaluation method besides accuracy.

Finally, the features we used to train the model may not be directly related to the author's happiness index, such as followers_count or retweet_count. For instance, the author released an ad without any strong emotion and got a large number of retweet_count since many people thought the ad is useful. In this case, the author's happiness index has nothing to do with the retweet_count feature.

# 8. CONCLUSION

In this project, we applied data mining and machine learning techniques on twitter data and built an efficient framework to

detect and predict twitter users' happiness index. A training corpus of tweets related to significant keywords were built based on our twitter data crawling strategy, and several machine learning algorithms were used to train the classifier of happiness index. Finally, we predicted happiness index for twitter users by their twitter timeline based on our classifiers with good performance, and get a line chart of happiness index for each user.

## 9. REFERENCES


[1] Roberts, Kirk, et al. "EmpaTweet: Annotating and Detecting Emotions on Twitter." LREC. Vol. 12. 2012.

[2] Mohammad, Saif M., and Tony Wenda Yang. "Tracking sentiment in mail: How genders differ on emotional axes." Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis. Association for Computational Linguistics, 2011.

[3] Kim S, Bak J Y, Oh A H. Do You Feel What I Feel? Social Aspects of Emotions in Twitter Conversations[C]//ICWSM. 2012.

[4] Bollen J, Mao H, Zeng X. Twitter mood predicts the stock market[J]. Journal of computational science, 2011, 2(1): 1-8.

[5] Bollen J, Mao H, Pepe A. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena[J]. ICWSM, 2011, 11: 450-453.

[6] Tweepy, http://www.tweepy.org/

[7] Joshua Roesslein. tweepy Documentation, Release 3.6.0. https://media.readthedocs.org/pdf/tweepy/latest/tweepy.pdf

[8] Structure of the status object of Tweepy, https://gist.github.com/dev-techmoe/ef676cdd03ac47ac503e85628277bf2

[9] Stanford sentiment lab: https://nlp.stanford.edu/sentiment/

[10] Thinknook twitter sentiment dataset: http://thinknook.com/wp-content/uploads/2012/09/Sentiment-Analysis-Dataset.zip

[11] Keras sequential model: https://keras.io/getting-started/sequential-model-guide/

[12] XGBoost, http://xgboost.readthedocs.io/en/latest/

[13] Imbalanced-learn, http://imbalanced-learn.org/

[14] Scikit-learn, http://scikit-learn.org/

[15] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[16] Plotly, https://plot.ly/python/