United States Airline Sentiment Analysis using Twitter data

Kunal Lalwani, Kyle Lemaire, Darshan Mange, Donglin Lao, Mark Ledesma

***Abstract*—PAST research has shown that real-time Twitter data can be used to predict market movement of securities and other financial instruments [1]. The goal of this project is 1) build a model for six major U.S. airlines that performs sentiment analysis on customer reviews so that the airlines can have fast and concise feedback, 2) Make recommendations on the most important aspect of services they could improve given customers complains. In this project, we performed multi-class classification using following Classifiers: a. Naive Bayes, b. SVM c. Logistic Regression d. Random Forest e. Decision Tree f. K-Nearest Neighbors, on the Twitter US Airline data set from Kaggle. Significant accuracy has achieved, which shows that our models are reliable for future prediction. Also, the accuracy of different models is compared, and results show that Random Forest is the best approach.**

I. Introduction

IN recent years, Twitter has become the de facto online customer service platform. Hence a company's reputation on the Twitter website is of high value and importance especially in case for airlines. This is assuming the fact that commuters and travelers usually tweet their experiences. In fact, study has shown that replying to tweets has revenue generating potential, imbibes mores satisfaction than other customer service portals, and perhaps most importantly, satisfied Twitter users spread the word. In this project, we use the tweets to learn about people's flight experiences and give airline companies suggestions on how to make their trip more enjoyable.

The data set contains about 15,000 tweets, collected from February 2015 on various airline reviews from Kaggle website's already available dataset. Every review is labeled as either positive, negative or neutral. First, we want to build a model to perform sentiment analysis on the data set. Second, more interestingly, we want to assign a reason to each negative response, such as late flight, lost luggage, etc. In our data set, about 80% of the negative reviews has a negative reason label, yet the rests are labeled as" can't tell". Our goal is to assign a label to this unspecified group. By knowing every review's negative reason, we can give specific suggestions to different airline companies on how to improve their service.

II. Background and Related Work

Nowadays, developing and testing different models for a natural language processing problem is an interesting and challenging task. However, due to the nature of the problem, the accuracy of sentiment analysis on single sentence like movie reviews never reaches above 80% for the past 7 years [1]. Looking at last year's project on twitter [2] their accuracy was 59.32% to 63.71%, depending on different models. In our project, we achieved near 20% more than their result, which is a significant improvement.

Since tweets texts are usually short and verbal, the same problem presents in our data set as well. However, even though the tweets are short, there are strong indicative words. Specific words can be used as indicators for spam/ham emails and achieve good test accuracy. Therefore, we believe that tweets review, without many negating negatives, can be predicted well using the frequency vector representation. To prove this, we will use 6 different models.

III. APPROACH

Twitter, for the past decade, is one of the most used microblogging websites in the world. With Twitter, people can easily express their opinions for the whole world to see, if person allows for their tweets to be public. Twitter has even allowed a more direct contact between businesses and the public, such that they can advertise or speak directly to customers. Also, the consumer can share their feelings towards a specific company, whether it has positive, negative, or neutral sentiments. This can either help the company, or hurt the company depending on the tweet. Since there are millions of users on twitter, and billions of tweets tweeted a day, it becomes difficult to people to search through the tweet to understand the sentiment of its consumers[3]. However, we can use machine learning to train and evaluate models to recognize the sentiment of a tweet, to help better and efficiently understand what the consumer wants.

A. Task

For our research, we are analyzing tweets related to various airlines to understand the people's sentiment when using their services. The dataset is used from Kaggle repository. We can see looking at the graphs below that most of our tweets have negative sentiments towards airlines with a 63%, while only 21% has neutral sentiment and 16% has a positive sentiment. Using these tweets, we first need to preprocess them to obtain the features we want that would affect the sentiment. Then we use a machine learning model, such as Naive Bayes, Random Forest to name a couple, and train the model using some of the preprocessed tweets.

Finally, using the testing dataset of preprocessed tweets, we evaluate and compare our results to the predefined sentiment and obtain the models accuracy, precision, and recall rate. Once complete, we can finally compare each model and see which is best suited for twitter sentiment analysis.

B. Data Set

The sentiment analysis labels are positive (20%), negative (60%), and neutral (20%). The negative reason labels are bad flight (7.45%), canceled flight (9.62%), customer services issues (39.8%), damaged luggage (0.84%), flight attendant complaints (6.05%), flight booking problem (6.19%), late flights (1.99%), long lines (19.97%), and lost luggage (8.23%).

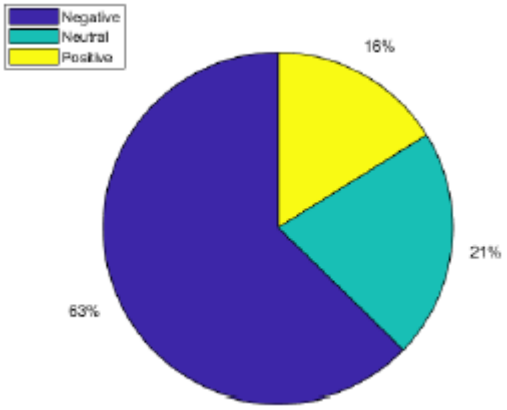


Figure 1: Pie chart for the percentage of sentiments in our dataset.

C. Preprocessing Steps

Before we can start training, the data must be preprocessed since there are words that are the same, but are written differently, such as WORD and word. They are indistinguishable to us, but to a computer they are not. We want to have as few, but necessary words as possible[4]. To do this, we use regular expressions (regex) on every tweet. While processing each tweet, we convert every character from upper case to lower case, convert URLs into the word URL, convert mentions or others username into the word AT USER, remove excess whitespace, remove all instances of '#' only keeping the word following it and then stripping the tweet. We also eliminate character repetition such that when someone uses the word LOOOL, LOOOOOL, or however many 'O's they can fit, it will all be converted to 'LOOL' since it the intended spelling of some words require two of the same characters in repetition, such as the word 'hello'. This does mean that 'LOL' and 'LOOL' are two different features, but it would be extremely tedious to go through each feature and differentiate between the words that do and do not require the repetition and remove the unnecessary features.

When obtaining our features, we first preprocess, as previously mentioned, and then we tokenize each word for all tweets. Going through each word, we remove each character repetition and punctuation. Finally, we compare each word in the tweet with stop words from the NLTK corpus, AT USER and URL. The stop words are words with no sentimental value, such as the word the, and therefore have no reason for being a feature. Also, the only features that are accepted are those that start with a letter so that there are no nonsense words. Finally, after checking every word, we get our features. For the tweets we want to train, we use the mentioned process and attach the appropriate sentiment value to it.

D. Dictionary

The dictionary is made based on the training data and all sentences are broken down into list of words: (1) Delete common words such as a, an, to, etc. with high frequency but little semantic usage. (2) Stem words, such as "thanks" and "thank" as one word. (3) Delete low frequency words to reduce the size of dictionary for calculation efficiency.

E. Bags of Words Representation (Feature Extraction)

As features in our dataset is words i.e. text so we need to represent in numbers which can be used as feature for various models. Bags of words is used to represent the text. It is way to extract features from text to use in the modelling. It describes the occurrence of words within a text/sentence.

Three ways to represent in numerical features:

• Tokenizing Gives an id for each possible token, for example white spaces and punctuation as token parameters.

• Counting the occurrences of words or tokens in each document.

• Normalizing and weighting the importance token and diminishing the weights of stop words.

Samples are defined as follows:

• Each individual token occurrence frequency is treated as a feature.

• Vector of the features or tokens.

Vectorization is the general process of turning a collection of text documents into numerical feature vectors. In our project we have used 2 grams of representation to increase the accuracy.

For example: "This book is not good" will have almost similar score value as "This is good". As each word is taken, to increase the probability we can take 2 words as single feature like "not good", "is good" will give us score value different. Score value is the number to which each word is assigned to. Stop words have less value compared to other values.

IV. MODELS

1. NAÏVE BAYES

One of many ways to solve the twitter sentiment analysis classification problem is by using Naïve Bayes. Naïve Bayes uses Bayes Theorem, which for our classification problem, gives us:

$$P(label \mid features) = \frac{P(label)\ x\ P(features \mid label)}{P(features)}$$

The algorithm assumes that all features are independent of one another and are hence naive. The Naive Bayes algorithm

from the Natural Language Toolkit (NLTK) normalizes the numerator for each label rather than computing P(features). In short, using the tweet as input, the algorithm will determine from each word which is the most appropriate label given what it knows already from training.

We test our tweets by comparing each word with those features obtained from our training set and determining using the Naïve Bayes method to obtain the most likely sentiment. The data used has more tweets with negative sentiment than either neutral or positive, so it's more likely that it will predict negative tweets correctly.

We can obviously see that with smaller test sizes, the predictions are more accurate, yet there's not much of a difference. We can also see that both positive and neutral recall only slightly vary as the test size increases. Negative recall, positive precision, and neutral precision get worse as the test size increases, while negative precision overall improves. Given the fact that most of the tweets have a ground truth label of negative, it makes sense that such

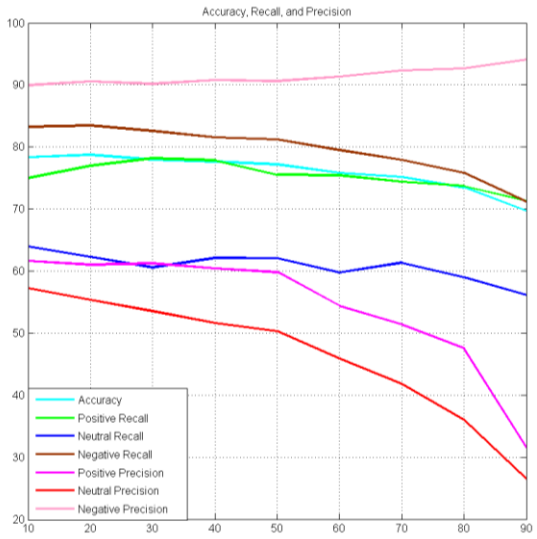


Figure 2: Comparing the results of different sized training and testing data to see how well it can predict the data correctly

events occur with smaller training data. Naive Bayes has some difficulty predicting neutral tweets while easily guessing negative tweets, though this is most likely due to there being a bias. If the dataset contained an approximately equal amount of positive, negative and neutral tweets, then surely the Naïve Bayes algorithm would have better predictive capabilities.

2. SUPPORT VECTOR MACHINE

What Support Vector Machine is- a supervised machine learning classifier that can implemented for regression analysis as well as binary classification. We are here using the classifier in our work to separate out tweets into positive or negative sentiments. We are using RBF Kernel in our project. SVM uses the same input and implementation package as Naive Bayes. We then calculate the accuracy of the SVM model. Following is the equation of the line that classifies our model.

$$K(u,v) = u^T v = \exp\left(\frac{-||u-v||^2}{2\sigma}\right)$$

The simple estimation of the decision boundaries to result as a function of the predictor variables is the main idea behind SVM. The input values are mapped on to one or other side of the boundary using this estimated line. The problem of overlapping classes, and the decision boundaries linearity is overcome by the SVM Classifier.

3. LOGISTIC REGRESSION

Logistic regression is implemented to solve classification problems where we want to predict the result to be either 0 or 1. We usually use a sigmoid function as defined in the equation:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

We can see that even if both Naïve Bayes and Logistic regression are linear classifiers, Naïve Bayes is a considered to be a generative classifier that will try to predict the likelihood term under the assumption of conditionally independent between features, however, this assumption less happened in the real word problems, but by contrast, Logistic regression is a discriminative classifier that use a Logistic function to get the likelihood directly. In addition to this, Logistic regression also covers the case of a binary dependent variable. As our dataset will be classified in a positive and negative category, it is expected to have better performance than Naïve Bayes, which can be clearly seen in the conclusion section

4. ADABOOSTING

Adaboosting is an ensemble technique which takes number of weak learners and attempts to create a strong classifier. It is used to boost the performance of decision trees. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. At each boosting iteration weights w (1), w (2), w(n) are applied to each of the training samples. Initial value of those weights is all set to w(I) = 1/N, the first step simply trains a weak learner on the original data. For each successive iteration, misclassification rate also known as error is calculated and the sample weights are modified based on that rate and the algorithm is applied again to the reweighted data. At each iteration weights of incorrect samples which we got from previous step, whereas weights of correctly classified samples are decreased. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence

n_estimators(Default Estimator)	Accuracy
100	74.487704918%
200	75.1024590164%
300	76.0587431694%
400	76.4344262295%
500	76.3319672131%

Table 1: Parameter Tuning

N_estimators controls the number of weak learners also known as decision stumps. Contribution of the weak learners in the final combination is controlled by learning rate. Base Estimator parameter is used to specify different weak learners.
N_estimators and the complexity of the base estimators are the main parameters to tune to obtain good results

N_estimator(Decision Tree)	Max_depth of tree	Accuracy
2	15	~73.66%
10	15	~74.28%
100	15	~70.41%
10	14	~74.11%
10	16	~74.86%

Table 2: Adaboost with Decision as base estimator:

As no of estimators is increased, accuracy also reaches till a point and it starts decreasing after a certain point in our case it is 300. As learning rate is varied, accuracy varies there is no pattern with the accuracy. Same is the case with max_depth. Major impact on the accuracy is due to N_estimators.

5. GRADIENT TREE BOOSTING:

Gradient tree boosting is dependent on three below factors.
Loss Function: It can be squared error or logarithmic loss
Weak learner for predictions: Decision trees are constructed in a greedy manner, choosing split points based on purity scores like Giny
An additive model: Decision trees are added in iterative manner and gradient descent procedure is used to minimize loss.
N_estimators: controls the number of weak learners
Learning_rate: it used to control the overfitting of the data.

N_estimators	Learning rate	Max depth	Accuracy
100	1.0	3	76.2978142077%
200	1.0	3	77.1516393443%
300	1.0	3	77.2540983607%
400	1.0	3	77.1857923497%
300	2.0	3	69.8428961749%
300	0.1	3	73.1315677889%
300	1.0	4	77.2540983607%
300	1.0	5	76.912568306%
100	1.0	4	76.6393442623%
100	1.0	5	75.9221311475%

Figure 1: Training Gradient Tree with different values of parameter

As no of estimators is increased, accuracy also reaches till a point and it starts decreasing after a certain point in our case it is 300. As learning rate is varied, accuracy varies there is no pattern with the accuracy. Same is the case with max_depth. Major impact on the accuracy is due to N_estimators.

6. DECISION TREE

The Decision tree classification approach is a technique that making classification predictions by carrying out a series of true or false decisions. The decision tree approach is the foundation for the implementation of Random forest. A Decision Tree is a flowchart-like tree structure, in which each internal node represents a test on an attribute and each branch represents an outcome of the test, and each leaf node represents a class.

7. STATISTICAL LANGUAGE MODEL

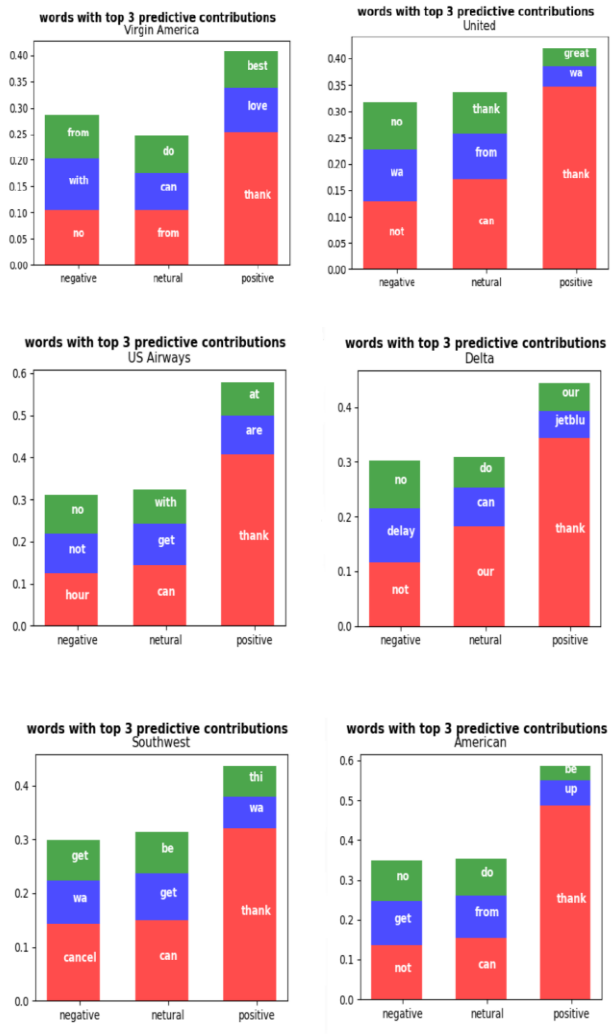
The Statistical Language Model, also known as SLM, is a model used to estimate the distribution of natural language. The model uses probability distribution to show how frequently a string is encountered and, based on the strings, it will make a prediction on what word would come next. The big thing with natural language is that it is always changing. Words and phrases used in a specific way can mean many different things. That is why we must look at the whole sentence structure and not just one word within a sentence to try and guess what the sentence is trying to convey. It assigns a probability to the whole sentence and also a probability to the likelihood of what word will show up after a given sequence of words are given.

The model begins with getting the tweets and breaking it up into words. The words are then analyzed to see what words are commonly used within the batch of tweets. The commonly used words are used as stopping points as the model grabs list of words from the tweets.

Word	Frequencies	Word	Frequencies
Have	1746	That	1722
To	8627	The	6030
I	5393	Flight	4762
A	4457	You	4082
For	3975	On	3763
And	3693	My	3265
Is	2808	In	2521
It	2494	Of	2099
Me	1908	Your	1899

Table 3: list of the most commonly used words within the tweets in the given file

The model removes the stopping words that have little to no effect to the sentence structure and it will make the data smaller to search through. This will also assign a probability to a sequence of words. The words that are important will have a higher probability next to ones that are not as important. The features part of the model will be a list of words that will get check to see what label it falls under. The labels are positive, negative and neutral. This model also uses MLE (maximum likelihood estimation) to calculate the probability of receiving a prediction based on what set of data it was given. Just knowing some predictions will help



the model get a good estimate for a bigger sample size, but only if the predictions are normally distributed. The scores I got after running the model 6 times for each alpha value:

Changing the alpha value to point 0.1:

98.6	93.54	94.52	94.63	95.21	94.9

Changing the alpha value to .5 and got:

97	88.17	92.45	92.54	87.33	89.88

Changing the alpha value to .01 and got:

98.8	94.78	95.02	95.59	96.66	95.66

Changing it once more to .0001 and got:

.98.8	94.96	95.18	96.22	97	95.85

The prediction value gets better when the change of alpha is smaller.

The next graphs are going to show the predictions for 6 major United States airlines. The graph will show the predictions the model found for the categories it predicted for the tweets that it analyzed and also for the top three words it used that helped with its prediction to what label it falls under. The graphs show the top 6 US airlines.

8. RANDOM FOREST

The random Forests regressor is implement for fitting many classifying decision trees on sub-samples of our data. This classifier is expected to improve the accuracy and control overfitting by averaging. The introduction of the Random Forest Algorithm was developed as an approach to be built over decision trees. We use the bagging approach as described in the approach section. From a dataset D, we build each time an eligible Decision Tree, having N instances and A attributes, we build a subset d. This d is sampled along the with replacement techniques for the training dataset.

9. NEURAL NETWORKS

We trained our neural networks for binary classification using the Twitter US Airline Sentiment dataset from Kaggle. Our models have the following equation for parameter norm penalty

$$\boldsymbol{J(\theta \mid X, y) = J(\theta \mid X, y) + \alpha \Omega(\theta)}$$
$$L1\ penalty\colon\ \boldsymbol{\Omega(\theta) = \Sigma|w|}$$
$$L2\ penalty\colon\ \boldsymbol{\Omega(\theta) = w^t w}$$

Where α is user defined to be 0.01 globally in all layers of our testing environment for both L1 and L2. This α values was selected randomly to be 0.01 since we were pressed for time and did not have a chance to find an optimal value for

Model #	*Method*	*Activation*	*Architecture*	*Accuracy%*
1	*RNN (LSTM Layer)*	*Tanh*	*25 layers,10 units*	*78.74*
2	*RNN (LSTM Layer)*	*Tanh*	*3 layers,100 units*	*91.02*
3	*RNN, dropout (0.2)*	*Tanh*	*25 layers, 100 units*	*78.74*
4	*RNN, dropout (0.2)*	*Tanh*	*3 layers, 100 units*	*90.23*
5	*RNN, dropout (0.4)*	*Tanh*	*25 layers, 100 units*	*78.74*
6	*RNN, dropout (0.4), L1_L2(0.01)*	*Tanh*	*3 layers, 100 units*	*90.03*
7	*RNN, dropout (0.4), L1(0.01)*	*Tanh*	*3 layers, 100 units*	*78.74*
8	*RNN, dropout (0.5)*	*ReLU*	*3 layers, 100 units*	*89.86*
9	*RNN, L1_ L2(0.01)*	*Tanh*	*3 layers, 100 units*	*90.11*
10	*RNN, dropout (0.5)*	*tanh*	*3 layers, 10 units*	*91.47*

Table 4: Displays the list of Neural Network architectures that were used in our experiment

our experiment. We defined the dropout rate to be either p = 0.2 and 0.4 in our network architectures. These values were chosen to be lower than 0.5 as dropout was first introduced by Srivastava et al.[5] because our sample size is quite small.

Our model takes the tweets and does some preprocessing to clean up the data and then extracts the features to feed into our Recurrent Neural Network Model (RNN) that uses Long Short-Term Memory layers (LSTM). We limited our features to 2000 components and train them on half of our dataset using a 10-fold cross validation technique. The hidden units of our feed forward neural network use a tanh activation function except for Model 8, where we used a ReLU function. Finally, our loss function uses a categorical cross entropy function using the following formula:

$$H(p,q) = \boldsymbol{\Sigma_x p(x)\, log\, q(x)}$$

We tested our training set on each model for 20 epochs, each epoch indicated one forward and backpropagation.

Our results from training of our dataset show that out of our 10 models, 4 of which were unable to learn and as a result predicted a negative sentiment for all our evaluation samples shown in table 1.

Our tests also found that for our dataset, deep RNNs and models with L1/L2 regularization were unable to learn the training set properly. Therefore, we set out to create shorter but deep networks[6]. We observed that the networks that were able to learn the model were able to achieve around 90% accuracy rate on our evaluation set[7]. The best model in terms of accuracy % was a 3x10 RNN with dropout p=0.5 capable of achieving 91.47% accuracy rate in our dataset.

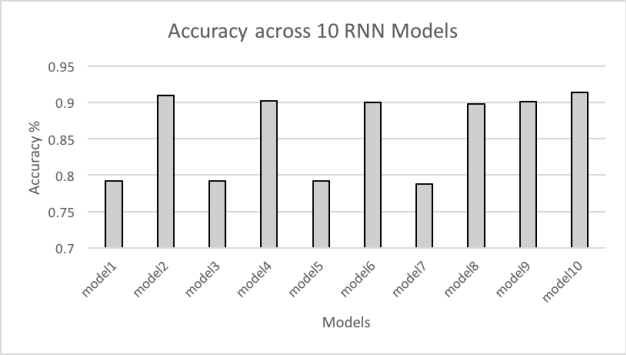


Figure 3: Comparison of the Accuracy of 10 different Neural Network Models

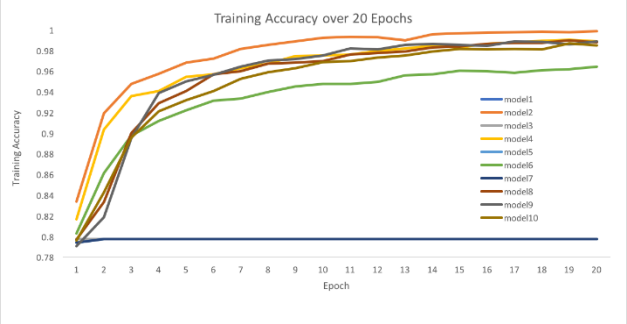


Figure 4: Training Accuracy of the 10 models over 20 epochs

Figure 5: Training Loss of the 10 models over 20 epochs

V. DISCUSSION

A general conclusion is that classifiers' results vary according to the dataset, however, it is noticeable that kNN, RF, and DT were almost better than other classifiers on most of the experiments. Another major conclusion is that feature selection (levels 1 and 2) almost enhanced every performed experiment on any dataset, except for rare cases where using the whole feature set was better (i.e. Naïve Bayes, and Adaboost on the "American" dataset).

There is no universal setting for the best Neural Network model. Therefore, in our study we set out to test multiple hyperparameters to achieve a higher performance on our dataset[8]. We explored different combinations of neural network architecture and explored the effects of deep versus wide networks as well as a combination of the three popular regularization techniques: L1, L2, and dropout. Based on our results, we found that a simple 3x10 RNN model with high dropout rate was best for our dataset.

Due to time constraints, we were limited in the number of models we were able to run and therefore did not investigate into searching for an optimal alpha value for L1, L2 and Dropout rate. To bypass this roadblock, we plan to utilize better hardware using Nvidia GPU processing power to speed up computations. Tensor Flow can run up to 50% faster on the latest generation of Nvidia GPUs compared to using CPU processing power and therefore we can speed up our training times dramatically in future tests.

VI. PRECISION

MODEL NAME	ACCURACY
RNN	0.915
Random Forest Classifier	0.809
Ada Boost Classifiers	0.785
Gradient Tree Boosting	0.765
Decision Tree Classifiers	0.757
Logistic Regression	0.645
SVM	0.645
Naïve Bayes	0.572

Table 5: Accuracy in the two-class dataset

VII. CONCLUSION

In this paper we have studied the sentiment analysis based

on the feedbacks of travelers regarding airline companies. Our proposed approach showed that both feature selection and over-sampling techniques are equally important about to boosting our results. The use of feature selection techniques has returned the best subset of features and reduced the computations needed to train our classifiers. Whereas, SMOTE has reduced the skewed distribution of the classes found in most of our smaller datasets without causing overfitting. Our results are a compelling evidence that the proposed model has high classification accuracy in predicting instances form the three classes (Positive, Negative, and Neutral).

As can be seen, some of the applied classifiers have outperformed the others. For example, Random Forest and Decision Tree have shown a high prediction level, and stability when applied on all datasets. While K-NN and Linear SVM have shown an acceptable level of performance regarding all the evaluation metrics. On the other hand, Kernel SVM has shown poor results in comparison with other classifiers

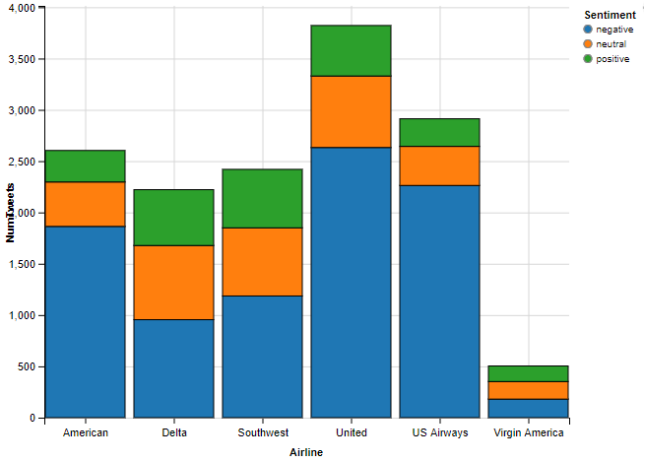


Figure 6: United had the most twitter commentary on it and US Airways had the highest fraction of negative twitter commentary. Virgin America had the least twitter commentary, but it also had the highest fraction of positive commentary

VIII. FUTURE WORK

Apart from measuring the performance of different models, in future, we plan to extract what kind of works exactly enables us to predict the emotions and sentiments of what people share as Tweets, more accurately. We initially had in mind to collect a dataset and depict it between a complaint or happiness emotion, but since our dataset was comparatively small, we planned to find the positive or the negative words mentioned in the Tweets which allowed us to improve our performance. One possible solution in order to improve our work will be to assign different weights to each positive/negative word in our dictionaries and categorize the tweet message according to their contents

IX. REFERENCES

[1] Esi Adeborna, Keng Siau, "An Approach To Sentiment Analysis-The Case of Airline Quality Rating," PACIS 2014 Proceedings. Paper 363.

[2] Hung T. Vo, Hai C. Lam, Duc Dung Nguyen, Nguyen Huynh Tuong, "Topic Classification and Sentiment Analysis for Vietnamese Education Survyey System", Asian Journal of Computer Science and Information Technology 6:3, May (2016).

[3] Soumi Sarkar, Taniya Seal, "Sentiment Analysis- An Objective View", Journal of Research, Volume 02, Issue 02, April 2016.

[4] Ann Devitt, Khurshid Ahmad, "Sentiment Analysis and The Use of Extrinsic Datasets in Evaluation," International Conference on Language Resources and Evaluation,2008.

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J Mach Learn Res 15, 1929-1958. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[6] Ian Goodfellow, Yoshua Bengio & Aaron Courville (2016). Deep Learning. MIT PressK. Elissa, "Title of paper if known," unpublished.

[7] Zagoruyko, S., & Komodakis, N. (2016). Wide Residual Networks. Procedings of the British Machine Vision Conference 2016. doi:10.5244/c.30.87

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90

[9] Scikit-learn: http://scikit-learn.org/stable

[10] http://www.itl.nist.gov/div898/handbook/apr/section4/apr412.html

CONTRIBUTORS

Kunal Lalwani

I was the Team Duck leader and worked on Decision Tree, Random Forest Classifier, Logistic Regression and Support Vector Machine Models. I started with pre-processing on the date-set to remove the noise, extract some features, and then the models were trained and implemented. The accuracy was calculated and compared by me for each model, resulting Random Forest as the best classifier.

Donglin Lao

I did all of work on the Neural Network Model and generated the data tables and figures and write up my findings on it. I tested 10 different model conditions including dropout, L1, L2 normalization, network width, network depth. I tested all the models in python using Keras to run on top of Tensorflow.

Mark Ledesma

I did the Statistical Language Model. I tested the tweets within the Model. I ran multiple test and change the values to see what results would be given during the test.

Darshan Mange

I did the feature extraction using bags of words representation. Implemented 2-gram model to extract feature. I trained AdaBoosting with Decision Tree Classifier as base estimator and Gradient Tree Boosting on the extracted features. nestimators, maxdepth and learning rate were the parameters tuned to get the best accuracy of the model.

Kyle Lemaire

I worked on the Naive Bayes Model from the Natural Language Toolkit. I tested and obtained the data for the accuracy, precision and recall rate on different testing data sizes. I compared which size was the most optimal size in terms of how well it was able to predict.