Multiple Model Analyses for Classifying Disaster Tweets

Ahmad Soni
University of the Witwatersrand
School of Computer Science
Johannesburg, South Africa
668564@students.wits.ac.za

Nasiem Ayob
University of the Witwatersrand
School of Computer Science
Johannesburg, South Africa
1825850@students.wits.ac.za

Nicholas Muir
University of the Witwatersrand
School of Computer Science
Johannesburg, South Africa
1399185@students.wits.ac.za

Simon Rosen
University of the Witwatersrand
School of Computer Science
Johannesburg, South Africa
1391995@students.wits.ac.za

Abstract

With the ubiquity of information on the internet, the ability to distinguish between what is relevant and what is not has become increasingly prevalent in interpreting such data. One of the more popular and potentially useful sources of real-time information available for use is that of social media websites. The problem we wish to address is how we distinguish between what information is useful and what is redundant or false when dealing with unmoderated posts made by people. We wish to solve this problem by applying various machine learning models to "tweets" from the popular social media application Twitter. Our model will attempt to predict if tweets related to disasters are real or fake and provide a categorization solution for future tweets' legitimacy.

Keywords
Logistic Regression, Social Media, Twitter, Prediction Model, LDA, NLP, LSTM, SVM, Bayesian, Machine Learning.

1 Introduction

With the rise of the internet and the ever increasing user numbers, this in turn has led to a huge amount of data being available to everyone at any given time. This is both incredibly valuable and dangerous at the same time. While we have access to all of this information, there is often no way to discern if the information is legitimate or false. This concern is particularly prominent in the existence of Social Media.

Social Media itself provides the unique opportunity for each individual user to have a voice or a say in literally every aspect of their lives. This can be incredibly helpful as well when dealing with things such as live reporting of events. This means that news about events can travel quickly and effectively throughout the world.

This leads us to our problem statement. How can we distinguish between which user posted stories are genuine or false? Particularly with regards to tweets about disasters.

By solving this problem we will be able to effectively gauge both current and future tweets for their validity. This should solve our problem statement and allow for less confusion when dealing with user posted stories particularly with regards to disasters.

The Social media platform we have decided to use for this is Twitter. This platform allows for users to post short stories to communicate just about anything at any time. These short messages are commonly referred to as "Tweets".

The models we are choosing to solve this problem are Logistic Regression,Long Short-Term Memory Recurrent Neural Network(LSTM),Support Vector Machine(SVM), Naive-bayes classifiers and Latent Dirichlet allocation(LDA).

The data we are using for this project was sourced from https://www.kaggle.com/c/nlp-getting-started/data

1.1 Previous Work

We previously researched this topic, however, we only investigated Logistic Regression and Long Short-Term Memory Recurrent Neural Network models. We are expanding on our research by adding Support Vector Machine , Naive-bayes and LDA classifiers.

1.2 Logistic Regression

Logistic Regression, similar to linear Regression, is a parametric classification model. What this means is that it functions by taking in a specified number of input "features" and outputs a categorical prediction. In our case, the features will be the words of the tweets and the output will be whether the tweet is a disaster tweet or not. The main difference between logistic and linear regression is, instead of using a linear function to fit our data we use a S-shaped curve, known as a Sigmoid. Logistic Regression can only be used for binary classification, so in this case, it works perfectly as our tweet is either legitimate or not.

When predicting a tweet, our model will return a probability between 0 and 1 which will give us the probability value of our tweet being legitimate or not. The scikit-learn package uses an ordinary least squares method to fit the Sigmoid curve when training the model.

1.3 Long Short-Term Memory Recurrent Neural Network:

A LSTM is a form or Recurrent Neural Network, which is used to process both single data points as well as sequences of data points. A LSTM Recurrent Neural Network is seen to mitigate the vanishing gradient problem that Recurrent Neural Networks face. This is done by Omitting certain data based off the previous layers output.

Since the the Long Short-Term Memory Recurrent Neural Network possesses the ability to learn from sequences of data makes it a plausible option for learning and predicting from text.

1.4 Naive Bayes

Naive Bayes classifiers are classifiers that function based on Bayes' Theorem. Bayesian classifiers are considered as a collection of algorithms which share the common principle of every feature being classified is independent of each other. Bayes theorem is given by the following:

$$P(A|B) = P(A)\frac{P(B|A)}{P(B)} \tag{1}$$

1.5 Support Vector Machine

A support vector machine is an algorithm that can perform both regression and classification tasks by finding hyperplanes in N-dimensional space. These hyperplanes separate data points into groups which can then be used for our classification and regression.

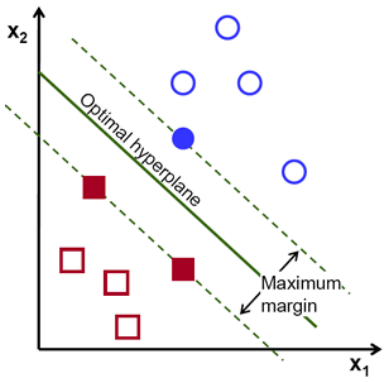

Figure 1: SVM Hyperplanes

1.6 LDA

An LDA model is a generative statistics model that makes observations based on unobserved groups of data in order to draw similarities between the groups. In an NLP setting, a LDA can discover topics in a collection of words and then classify each word within the collection based on their relevancy to other topics. This can be used to classify our disaster tweets into ones which are genuine and those which are not.

1.7 Optimizers

Where Stochastic gradient descent (SGD) maintains a single learning rate for Recurrent Neural Networks, the Adam optimizer combines two extensions of SGD namely Adaptive Gradient Algorithm and Root Mean Square Propagation. This allows for the use of the uncentered variance to be used with the mean, rather than only using the mean for parameter adaption.

1.8 Loss functions

Within the Tenserflow (Abadi et al., 2015) package Keras (Chollet et al., 2015). 2015], there are several loss functions that can be used in the model

for gradient calculations. These loss functions calculate the gradient using different methods such as the Poison loss which can calculate the loss of the dataset with regards to a Poisson distribution.

1.9 Literature Review

This problem has been addressed by other researchers in the past. Namely the paper by (Stowe et al., 2016) where a SVM model was used to classify. They were able to classify relevant tweets with high precision, however, they noted that some tweets do not contain enough information to classify and as such they would require additional work to provide a more fine tuned classification.

In a paper by (Ben Ltaifa et al., 2018) the traditional Bag of Words approach was replaced with a semantic approach, making use of Wordnet and concepts from DBpedia which could substantially improve the quality of tweet categorization when paired with another model.

2 Methodology

2.1 Data Preprocessing

2.1.1 Punctuation and Lowercase

Initially the punctuation marks '[,?]' are removed from the texts and converted to lowercase. This is because the presence of both would cause our algorithm to read what is essentially the same word as two different words. For example, 'because?' , 'because' and 'Because' would all be classified as different words when they are essentially the same.

2.1.2 Tokenization

This is the process of breaking up a sequence of strings into words which are known as tokens. We used 'Tweet Tokenizer' from python's nltk library to achieve this. It works by breaking up each sentence or tweet into individual words. This step is required for the lemmatization stage later in the algorithm.

2.1.3 Stopwords Removal

Commonly used words are created and extended using python's nltk. These words were then removed altogether with unnecessary remaining characters to reduce any noise that may occur in our data.

2.1.4 Lemmatization

This is the process of categorizing the words we have from the tokenization step. We convert each of the words into the different parts of speech, namely nouns, verbs, adverbs and adjectives. This is done to reduce our vocabulary and dimensionality for better speed and efficiency.

3 Base Model Implementations

3.1 Logistic Regression, Naive-Bayes Classifier and SVM

We trained and fitted a logistic regression model from python's sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

The data to be used in the LRM/Bayesian and SVM models were pre-processed as per the above preprocessing section and modeled very similarly as opposed the LSTM.

Since these model requires an input of numerical values and not strings , we used the unique words to create what is known as a bag of words(BOW). A bag of words is a representation of text that counts the total number of words that appear in each instance or in this case tweet. To create the bag of words, we used python's count_vectorization function.

The generated bag of words is then used to train our model.

3.2 LSTM

The initial model created consists of The input layer where each word is embedded into a vector for input into the model, this embedding is where each word is represented as a vector of integers. The model then consists of two Dropout layers, one LSTM layer and One dense layer as the output layer of size one. This model will be adapted to test for different accuracy's with varying dropout rates, LSTM layer sizes, different Optimizers and different Loss functions . Dropout layers are used to prevent over-fitting of data, this is done by randomly setting input units to 0 at a frequency that is varied in this experiment (Chollet et al., 2015).
The Model is then used to test the accuracy of the model using different Optimizers and Loss functions. The loss functions tested are Binary Cross-Entropy, Mean Squared Error and Mean Absolute Error, and the Optimizers are ADAM and Stochastic gradient descent. Each combination of Optimizer to Loss Function is tested for twenty epochs and each implementation is then bench-marked and saved in a CSV file pertaining to the Optimizer and

loss function name.
Different sizes of the Word Vector is then tested for accuracy of the model. Using the Model and varying the Word Vector model size to train and test the model, after which the accuracy is saved into a CSV file pertaining to the vector length. Testing for different layer sizes for the LSTM is also covered, the sizes of the LSTM layer tested are : 128,256,512. These accuracy's are saved into a CSV file named by the layer size.
The Dropout layer rate was tested as to how the accuracy was affected, this was achieved by varying the dropout rate for the Model. The dropout rates tested are : 0, 0.1, 0.2, 0.3 Accuracy's are then saved into a CSV file named by the dropout rate. All saved CSV files are used in visualization of the outcomes for each test applied above. Each Model trained is finally tested on unseen data, both the loss function localised to the model as well as the accuracy of each model. This data as well as the data stored for the validation and training is compared and discussed.

4 Results

To evaluate the performance of our model, we used python's confusion matrix function. The confusion_matrix helped us determine the accuracy of the models by displaying the total number of correct and incorrect predictions made by our model.

The positive predictions are weighed against the negative predictions to give us a final accuracy score for our models.

4.1 Logistic Regression

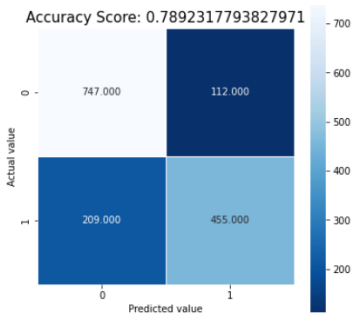

Figure 2: Confusion Matrix for LR model used

4.2 Naive-Bayes Classifier

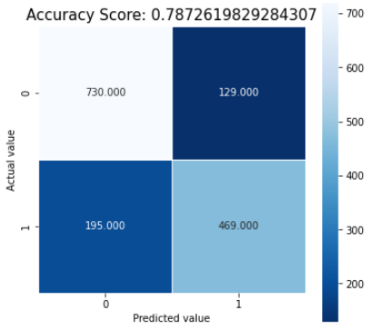

Figure 3: Confusion Matrix for Naive-Bayes model used

4.3 SVM

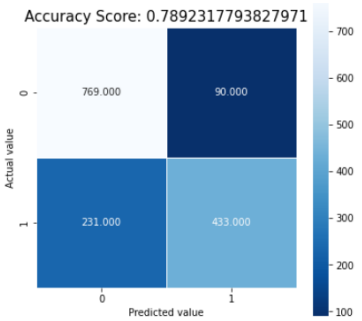

Figure 4: Confusion Matrix for SVM model used

4.4 LSTM

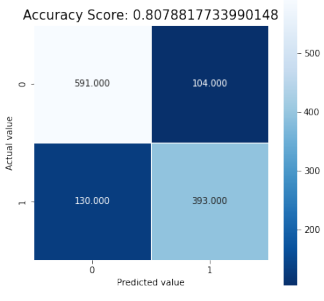

Figure 5: Confusion Matrix for LSTM model used

4.5 LDA

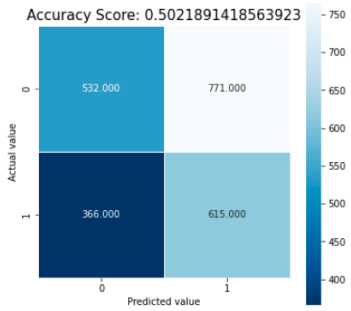

Figure 6: Confusion Matrix for LDA model used

5 Analysis

The results of our algorithms confusion matrices can be represented by the following table

Models	Accuracy Score
Logistic Regression	0.79
Naive-Bayes	0.79
SVM	0.79
LSTM	0.81
LDA	0.50

We can see that almost all of our models display a high and similar accuracy score with the exception of the LDA model. The logistic regression, naive-bayes and SVM models all show an accuracy score of 0.79%. The LSTM model has a slightly higher accuracy at 0.81%. Our LDA model however under-performed with an accuracy of only 0.50% making it too inaccurate to consider as a usable model.

Unlike the previous models, the LDA model does not have an in-built accuracy function, we had to figure out a way to manually compute the model's accuracy. We got creative and decided to use a tabular method from https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/. This method not only allowed us to retrieve the dominant topic for each text/tweet but also helped us to represent it in a tabular fashion. By doing so, we were able to extract the now predicted topics("dominant topics") per tweet and use the actual category of the tweet to calculate the model's accuracy.

6 Impact and Conclusion

Our research has shown that we can use multiple different models to solve the problem of correctly classifying disaster tweets based on their content. The results of these models show that we could viably write algorithms which can differentiate between genuine and fake information on the internet, which will become an increasingly prevalent problem as our online presences continue to grow.

The success of this project shows that we could use similar models to solve similar problems in classifying data on social media and the internet. It could technically be applied to any problem that has binary outcomes, such as classification of genuine and fake news online.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ibtihel Ben Ltaifa, Lobna Hlaoua, and Ben Maher. 2018. A semantic approach for tweet categorization. *Procedia Computer Science*, 126:335–344.

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Kevin Stowe, Michael Paul, Martha Palmer, Leysia Palen, and Kenneth Anderson. 2016. Identifying and categorizing disaster-related tweets. pages 1–6.