FINAL REPORT

Classifying and Identifying patterns in Fake Tweets

By Dhanya Lakshmi (dl998), Rashmi Sinha (rs2584), Enpei Jesse Yang (ejy26)

I. ABSTRACT

We used various machine learning techniques to identify fake tweets. We used Twitter APIs to generate our data to ensure that our data replicates the real world scenario. We relied on the fact that parody accounts tweet news which are generally fake, and the legitimate news accounts tweet real news. To ensure that we had reliable and pre-labeled data from external sources, we utilized the Liar and Kaggle Datasets (described below). In doing so, we obtained a mixture of real world tweets and news datasets, which was then preprocessed using techniques such as limiting sentences to the first *n* words, and incorporated the sentiment of the tweets in order to improve the performance. We observed that the LSTM model performed the best to identify fake tweets with relatively higher accuracy and sensitivity.

II. MOTIVATION

Lately, the amount of false information in tweets has grown exponentially, and has been aggravated by recent events such as COVID-19 and the election. This false information impacts the lives of people in an adverse fashion, causing them to believe information not backed by legitimate sources - for example, tweets that promote false information about causes and cures for COVID-19. For our analysis we used techniques such as CNNs, SVMs, LSTMs, and Transformers to build models to identify fake tweets.

III. METHOD

Data sources

The selection of databases was an important step for our project. Since our objective in this project was to identify fake tweets, i.e., tweets that contained misinformation about real-world events, we were interested in creating datasets with different types of data. This included pre-labeled statements from politicians and news organizations, and tweets we extracted using Twitter APIs that contained fake news tweets as well as real news. To ensure that we were testing on data that was as close to real-life as possible, we also extracted tweets over different days to make up the training and test sets.
We used the following datasets to train and test our models.

1. Liar dataset
2. Kaggle Dataset
3. Twitter APIs to extract tweets from parody accounts and legitimate news accounts. We applied for Twitter API keys and extracted tweets from the twitter accounts of legitimate news websites (e.g. - NYtimes, WSJ) which we considered real news. For fake news, we collected tweets from parody accounts (e.g. - TheFakeCNN, BloombrgNewsish). We assigned the labels accordingly and used the same to train our data.

Data pre-processing

For data preprocessing, in the Liar Dataset, we considered the labels 'mostly true', 'true' as Real news. We considered the labels 'false', 'half true','barely true' and 'pants on fire' as Fake news. In this way, we converted the labels to binary classification.

The following preprocessing steps were taken: converting all words to lowercase, stripping punctuation from the sentences, removing stop words, using Tokenizer to vectorize the text corpus to turn tweets into a sequence of integers, lemmatization of words using Portstemmer to retain their basic meaning, and removing links and special characters in text.

Data Selection and Evaluation Metrics

We followed a principal approach of iteration in choosing various models. The Liar Dataset was already split into the train set, dev set and test set, which we used directly. For the data retrieved from Twitter APIs and Kaggle, we used a split of 0.7 for train set to dev set. For our test set, we retrieved the data from the Twitter APIs using a different set of news accounts and parody accounts a few weeks after getting the training dataset so that we could evaluate our models on the future dataset to reflect the real world scenario and measure the temporal sensitivity of our model. Our main evaluation metric was the F1

score and accuracy. Also as the objective of this problem is more focussed on identifying Fake tweets, we are fine with having some false positives. Hence we want our models to have high sensitivity..

Convoluted Neural network

The distributional hypothesis states that words which have the same neighboring words are usually semantically similar (link). We intend to use a neural network with embeddings so that we can identify similarities in words that make up fake tweets or news, and use these similarities to predict whether new tweets we receive are fake or not.

Recurrent Neural Networks

Similar to above, using recurrent neural networks, we transform our words into vectors which then are processed in sequence. Where we differ from ordinary RNNs is utilization of an LSTM or Long-Short Term Memory. The idea is to reduce the effects of short-term memory on the recurrent neural network by adding a "memory" component of the network via cell states. This is accomplished with mechanisms called gates that control the flow of information, utilize activation functions to "forget" information that isn't required (in particular the sigmoid function).

We explored this machine learning technique because it could allow the models to keep relevant information about fake tweets during training whilst forgetting irrelevant data and could improve performance for fake tweet detection. To do so, we used a neural network with the following layers: (embedding, dropout layer, lstm, dense layer of 1 neuron).

Support Vector Machines

We are also exploring support vector machines for classification in an attempt to find an algorithm that can find the decision boundary that separates our two classes (Fake, Real) and maximizes the margin of the boundary.

We utilize SVM in both the linear and non-linear case in our experiments. For the linear case, we use the SVMs to find the optimal line to separate the classes or rather separate hyperplanes to classify the dataset correctly. This is denoted by a hyperplane that is n-1 dimension for the dataset in n dimensional space.

Since it is likely that our dataset is not going to be linearly separable, we relax the conditions for maximizing the margin and use a soft margin that is trying to find the line or hyperplane to separate the dataset whilst tolerating some misclassification errors and utilizing the kernel trick to find the non-linear decision boundary for our dataset (in our experiments we use the radial basis function kernel for our SVC).

BERT (Bidirectional Encoder Representations from Transformers)

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

IV. EXPERIMENTATION

Initial Baseline

To get an initial baseline, we ran logistic regression on our training data.

Logistic Regression

Description

For a logistic regression model we used the L2 norm on our train set with tf-idf vectorization. The training data was preprocessed as similarly to the linear svc and the model was tested on the twitter test set.

Train/Test Accuracy

The accuracy of this model on our twitter test set was 0.50. This was after a much higher accuracy during training and validation, that suggests some overfitting.

Evaluation Metrics

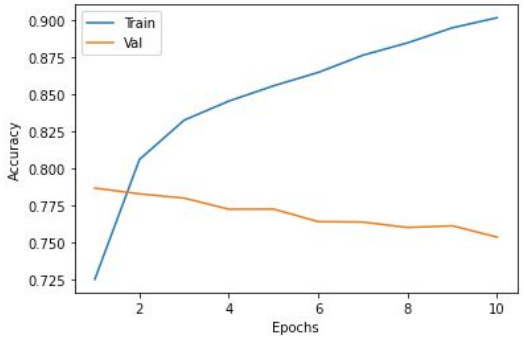
Sensitivity	**0.28**
Accuracy	**0.5**
F1 score (Real)	**0.39**
F1 score (Fake)	**0.58**

	Predicted Label		
Real Label		**1 (Fake tweet)**	**0 (Real tweet)**
	1 (Fake tweet)	**2891 (0.62)**	**1754 (0.38)**
	0 (Real tweet)	**7261 (0.54)**	**6136 (0.46)**

Experiments

1. LSTM

Description
We used a neural network with the following layers: (embeddin
g, dropout layer, lstm, dense layer of 1 neuron). We trained the models with 2 different features, the text of the tweet and the sentiment of the sentence.
Train/Test Accuracy
With a train-test split of 0.33 on the train set, we achieved an accuracy of 0.7536 on the validation set, and a train accuracy of 0.9. However, on our test set, our F1 score decreased significantly to 0.6314. Both accuracies are not satisfactory, but the decrease does indicate that our trained model was less performant on our live test twitter dataset, which could be indicative of overfitting or our liar training polluting our training.
Learning Curve



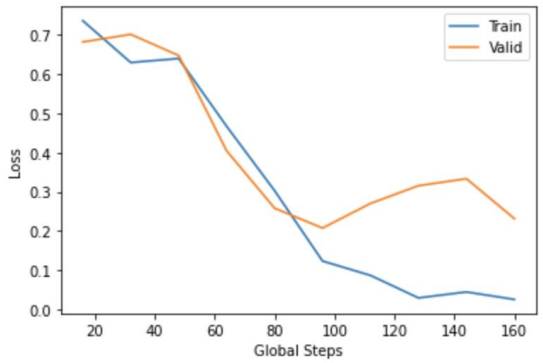
Evaluation Metrics

Sensitivity	**0.79**
Accuracy	**0.75**
F1 score (Real)	**0.69**
F1 score (Fake)	**0.79**

	Predicted Label		
Real Label		**1 (Fake tweet)**	**0 (Real tweet)**
	1 (Fake tweet)	**6076 (0.82)**	**1335 (0.18)**
	0 (Real tweet)	**1827 (0.34)**	**3597 (0.66)**

2. Transformers (BERT)

Description
We are using the “bert-base-uncased” version of BERT, which is a smaller model pretrained on lower-cased English text (with 12-layer, 768-hidden, 12-heads, 110M parameters). We further fine tuned this to classify fake tweets using the ‘Adam’ optimizer, the BinaryCrossEntropy loss function, and a suitable learning rate for 5 epochs. We also made sure that the output was passed through Sigmoid before calculating the loss. During training, we also saved the model each time the validation loss decreased and hence, found the model with the lowest validation loss.
Train/Test Accuracy
We used Kaggle Dataset and Twitter dataset to train this model. We obtained an accuracy of 99.8 % on the training set and 90.55 % on the dev set.
Loss Curve -



Evaluation Metrics

Sensitivity	**0.27**
Accuracy	**0.48**
F1 score (Real)	**0.55**
F1 score (Fake)	**0.39**

		Predicted Label	
		1 (Fake tweet)	**0 (Real tweet)**
Real Label	**1 (Fake tweet)**	**3019 (0.70)**	**1256 (0.30)**
	0 (Real tweet)	**8067 (0.58)**	**5699 (0.42)**

3. Convolutional Neural Network using Pre Trained Word Embeddings

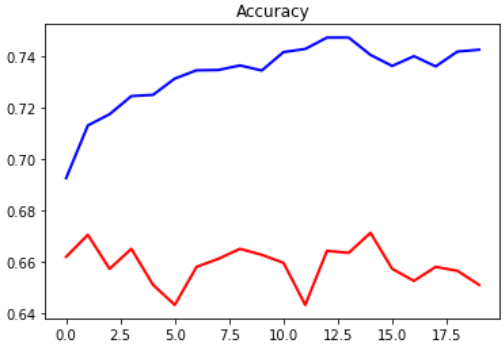
Description

We used a neural network with pre-trained word embeddings - gloVe and FastText to classify the tweets. We trained the model using 3 different features - the text of the tweet itself, the sentiment of the sentence, and the context/subject of the tweet. The final neural network had the following layers - Embedding, Dropout, Conv1d, Max Pooling 1d, Conv1d, Global Max Pooling 1d, Dense layer of 32 neurons, Dropout, and finally, Dense layer of 1 neuron.

Train/Test Accuracy

The accuracy on the training set increased over the number of epochs, and at the end of 20 epochs, was **0.7426.** On the other hand, the validation accuracy fluctuated and at the end of 20 epochs, reached **0.6511** with a highest accuracy of 0.6713.

Learning Curve



Evaluation Metrics

Sensitivity	**0.64**
Accuracy	**0.65**
F1 score	**0.75**

		Predicted Label	
		1 (Fake tweet)	**0 (Real tweet)**
Real Label	**1 (Fake tweet)**	**1528 (0.93)**	**108 (0.07)**
	0 (Real tweet)	**834 (0.92)**	**64 (0.08)**

4. **Linear SVC**

Description

For a linear SVC model we used tf-idf vectorization. It was trained on our combined liar and custom twitter dataset. The preprocessing we utilized was similar to what we described in our preprocessing techniques above. To create the model we used sklearn's SVC with a linear kernel.

Train/Test Accuracy

The accuracy of this model on our twitter test set was **0.39** . Which is incredibly low, this can be seen in the confusion matrix below, where the model had a lot of identifying false labels for our test set, seen with the low precision score for label 1.

Evaluation Metrics

Sensitivity	**0.72**
Accuracy	**0.39**
F1 score (Real)	**0.41**
F1 score (Fake)	**0.37**

		Predicted Label	
Real Label		**1 (Fake tweet)**	**0 (Real tweet)**
	1 (Fake tweet)	**3807 (0.28)**	**9590 (0.72)**
	0 (Real tweet)	**1436 (0.31)**	**3209 (0.69)**

V. CONTEXT

Throughout our experiments, we tested different ways of identifying fake tweets using a mixture of data sources. There aren't many previous work references with respect to identifying fake tweets available. To build the CNN with embeddings, we first used similar models as the ones previously used in different research papers with high levels of accuracies. Similarly, we initially used an off-the-shelf model for the LSTM implementation. However, due to the difference in data sources, we were unable to replicate the accuracies, and had to modify the models by adding/removing layers so as to find the best combination for our objective.

VI. CONCLUSION

From analyzing our experiment results, we found that certain strategies, like shortening the length of the text, improved our models. We believe that the nature of our training data may contribute to our low accuracy, namely the fact that we mixed the Liar dataset with tweet data harvested from legitimate and fake news accounts. The above technique likely worked well because each tweet has a limit of two hundred and eighty characters, while the liar dataset has no such restriction. In addition, there are other key differences between politician statements in the Liar dataset and news organization tweets, such as the prevalence of the words 'Breaking News' in the latter and the differences in sentence structure, grammar, and tone. The tweets tended to be much shorter, used banal language, and were less grammatically correct than the text found in the Liar dataset. With regards to the CNN's relatively high F-1 score and comparable accuracy, we suspect it performed well due to the large amount of common words in the data.

Error analysis of the different models revealed that different models had different measures of specificity and sensitivity. As we approached the problem from a practical standpoint, we considered two use cases where the preferred model might be different. For use case one, a company or organization that would like a function to determine or flag a given tweet is fake, a high sensitivity may be preferred as the model would have a higher proportion of true positives correctly identified, i.e. the function does a good job at flagging fake tweets, even if it may have some false positives. The second use case, perhaps in the context of a news organization verifying information, a higher specificity may be preferred, i.e. a higher proportion of true negatives. This is because for a news organization that necessarily relies on credibility, it may be more costly for the organization to mislabel news that is factually correct, rather than to flag information that might not be correct. For the first use case, our experiments reveal that of the models tested, LSTM (high sensitivity and high accuracy) would be the best model. In contrast, for the second use case, our experiments reveal that the Transformers (BERT) model would perform better.

Overall we determined that while one can explore additional features in a practical problem like this, i.e., adding in the results of sentiment analysis as another feature of our data, it is more important to understand the underlying data especially data that is not derived from a toy dataset.