FAKE NEWS DETECTION

CSPE 21 : Machine Learning Project
Semester : V
Date : 29.10.2018

Aishwarya Sarangu
106116001
Department of CSE
NIT Trichy
aishwaryasarangu@gmail.com

Harshitha Sanjeev
106116083
Department of CSE
NIT Trichy
harshi.sanjeev@gmail.com

ABSTRACT

The goal of this paper is to identify the authenticity of a news source and this is implemented by using Natural Language Processing techniques. We use labeled real and fake new articles to build a classifier that can make decisions about information based on the content from the corpus. We use four different classification models and analyse the results.

Upon implementation, we find that the best performing model is the LSTM.

The model focuses on identifying fake news sources, based on multiple articles originating from a source. Once a source is labeled as a producer of fake news, we can predict with high confidence that any future articles from that source will also be fake news. Focusing on sources widens the article misclassification tolerance, because we then have multiple data points coming from each source.

INTRODUCTION

Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the widespread of "fake news", i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention.

Fake news is increasingly becoming a menace to our society. It is typically generated for commercial interests to attract viewers and collect advertising revenue. However, people and groups with potentially malicious agendas have been known to initiate fake news in order to influence events and policies around the world. It is also believed that circulation of fake news had material impact on the outcome of the 2016 US Presidential Election. For these purposes, we have attempted to build a model that detects fake news with a fair accuracy.

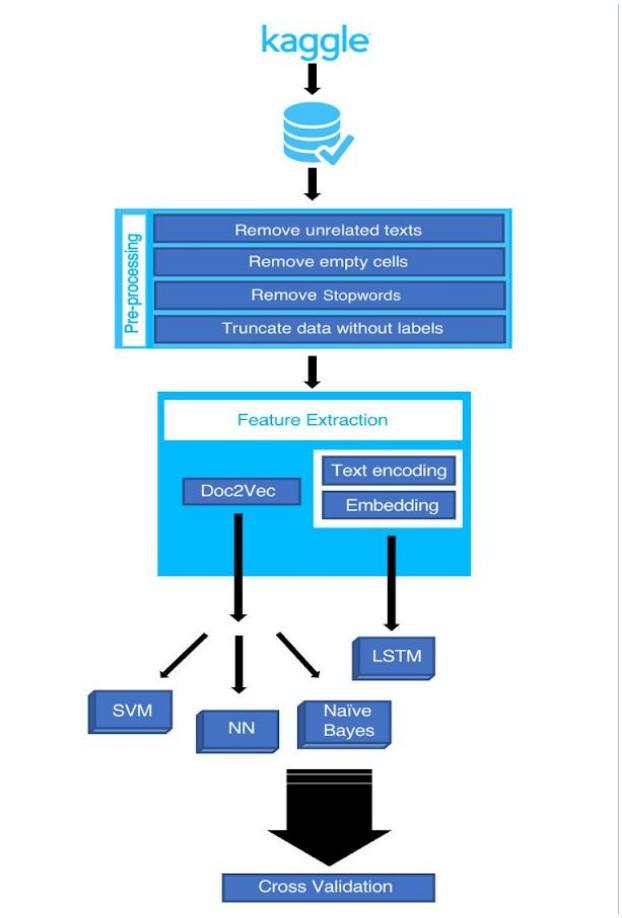
DATA

The datasets used for this project were drawn from Kaggle. The training dataset has about 16600 rows of data from various articles on the internet. The data was pre-processed, in order to train our models.

The full training dataset has the following attributes:

1. id: unique id for every news article
2. title: the title of the corresponding news article
3. author: author of the news article
4. text: the text of the article
5. label: a label that marks the article as potentially unreliable or reliable
    - 1: unreliable
    - 0: reliable

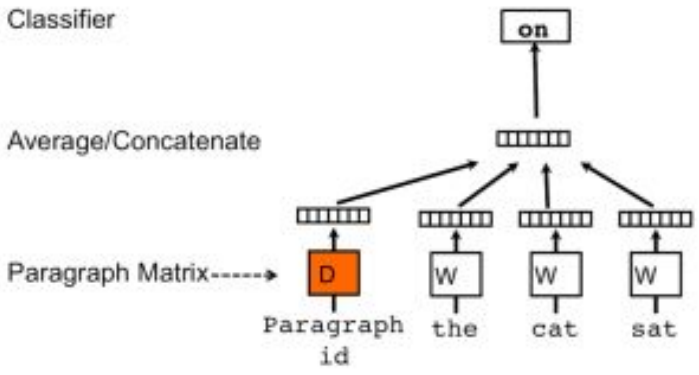
WORKFLOW



FEATURE EXTRACTION AND PREPROCESSING

The embeddings used for the majority of this modelling are generated using the **Doc2Vec** model. Doc2vec is an unsupervised algorithm to generate vectors for a sentence or paragraphs or documents. Doc2vec is an extension of **word2vec**, which represents documents by combining the vectors of the individual words, but in doing so it loses all word order information. Doc2Vec expands on Word2Vec by adding a ”document vector” to the output representation, which contains some information about the document as a whole, and allows the model to learn some information about word order.

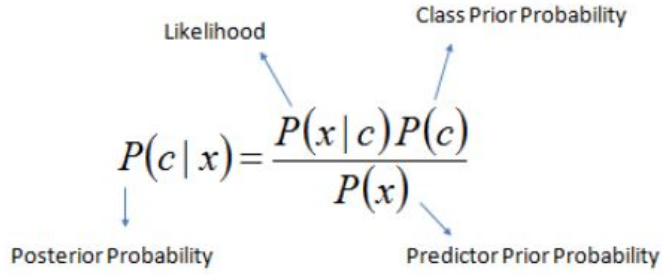
Before applying Doc2Vec, some basic pre-processing of the data is performed. This includes removing stop words, deleting special characters and punctuation, and converting all text to lowercase. This produces a comma-separated list of words, which can be input into the Doc2Vec algorithm to produce an 300-length embedding vector for each article.



MODELS

NAIVE BAYES

This is a simple yet powerful classification model that works very well. It uses probabilities of the elements belonging to each class to form a prediction. The underlying assumption in the Naïve Bayes model is that the probabilities of an attribute belonging to a class is independent of the other attributes of that class. Hence the name 'Naive'.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

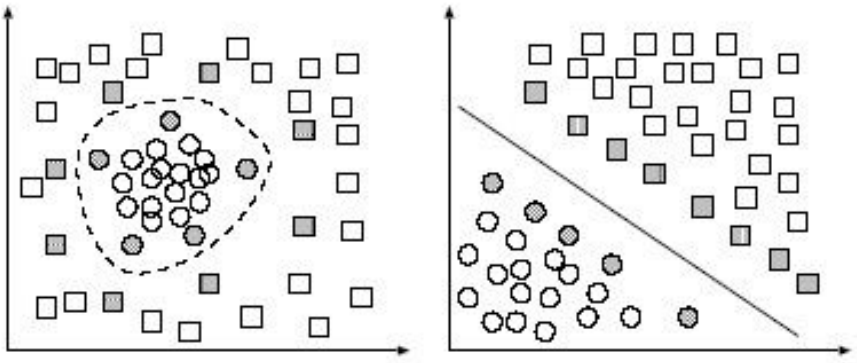
Parameter estimation for naive Bayes models uses the method of maximum likelihood. The advantage here is that it requires only a small amount of training data to estimate the parameters.

SVM

Support Vector Machines are machine learning models that perform supervised learning on data for classification and regression.
Here, the Radial Basis Function kernel is used in this project. The reason to use this kernel is that two Doc2Vec feature vectors will be close to each other if their corresponding documents are similar, so the distance computed by the kernel function should still represent the original distance. The Radial Basis Function is:

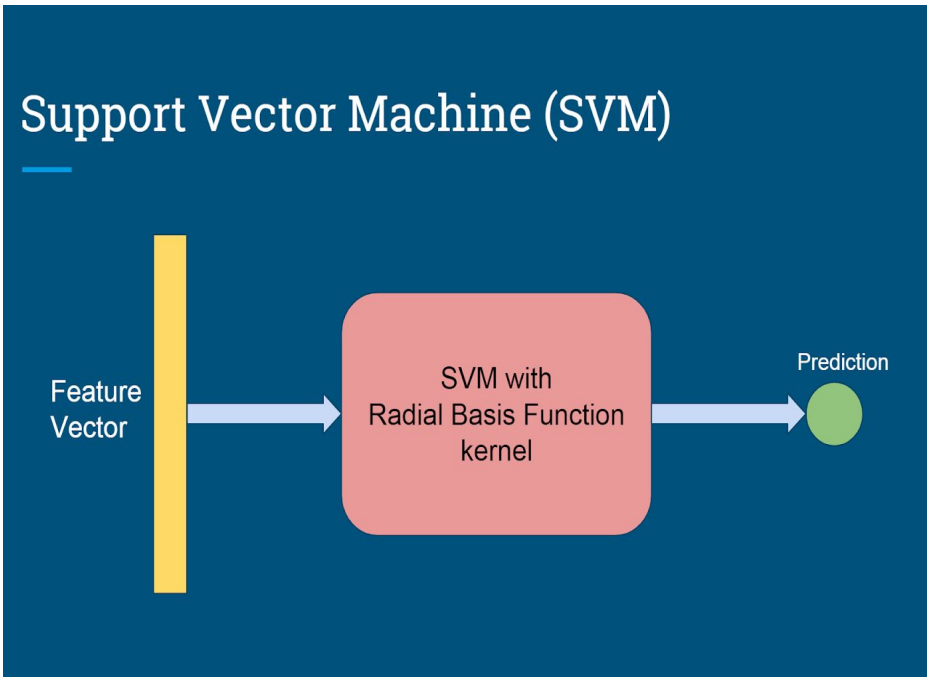
$$K(x, x0) = \exp -(||x - x\,0||^2 / 2\sigma^2)$$

It correctly represents the relationship and it is a common kernel for SVM. The main idea of the SVM is to separate different classes of data by the widest "street".



(a) Radial Basis Function (b) RBF mapping

Separable classification with Radial Basis kernel functions in different space. Left: original space. Right: feature space.



FEED-FORWARD NEURAL NETWORK

We implemented two feed-forward neural network models, one using Tensorflow and one using Keras.

Neural networks are commonly used in modern NLP applications and our neural network implementations use three hidden layers.

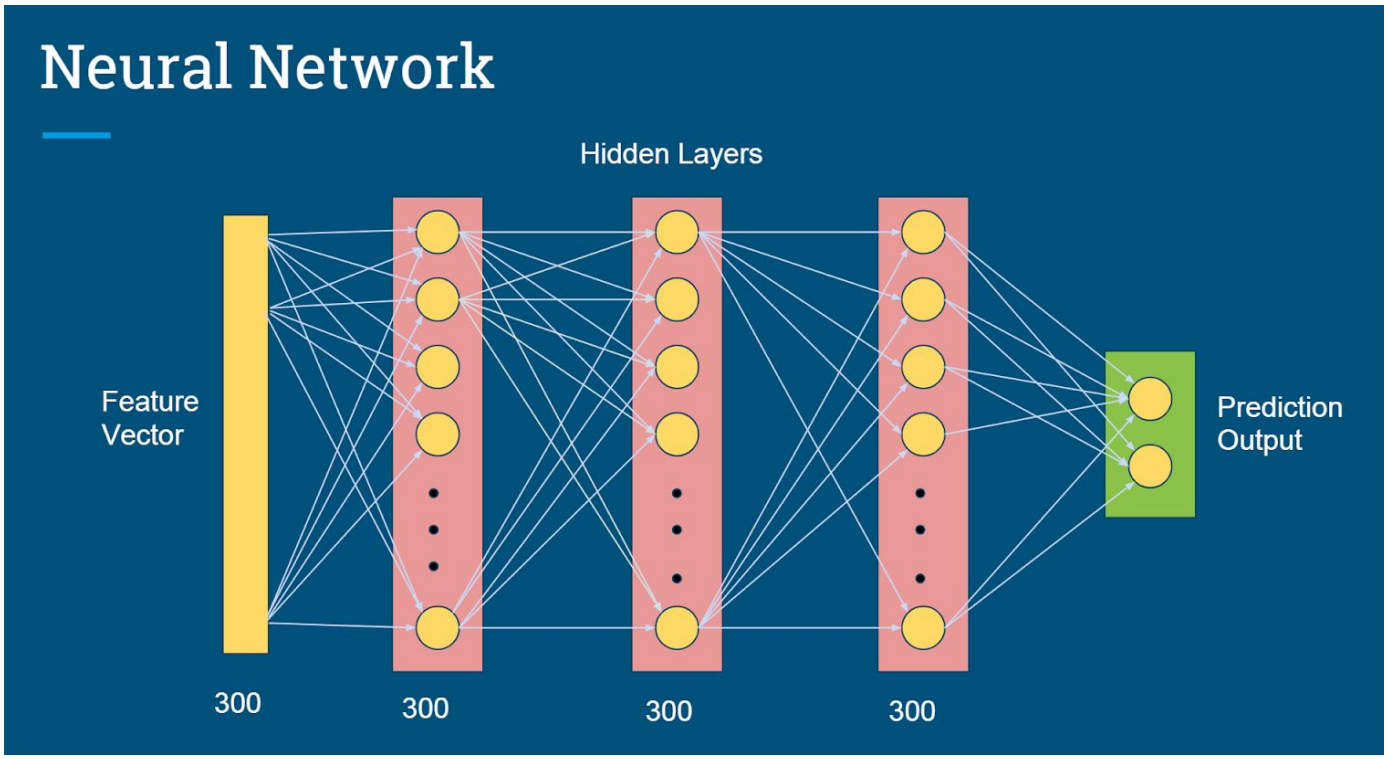
In the Tensorflow implementation, all layers had 300 neurons each

In the Keras implementation we used layers of size 256, 256, and 80, interspersed with dropout layers to avoid overfitting.

For our activation function, we chose the Rectified Linear Unit (ReLU), which has been found to perform well in NLP applications.

This has a fixed-size input $x \in \mathbf{R}^{1\times 300}$

$$h_1 = \text{ReLU}(W_1 x + b_1)$$
$$h_2 = \text{ReLU}(W_2 h_1 + b_2)$$
$$y = \text{Logits}(W_3 h_2 + b_3)$$



A simple neural network with 3 hidden layers

LONG SHORT TERM MEMORY

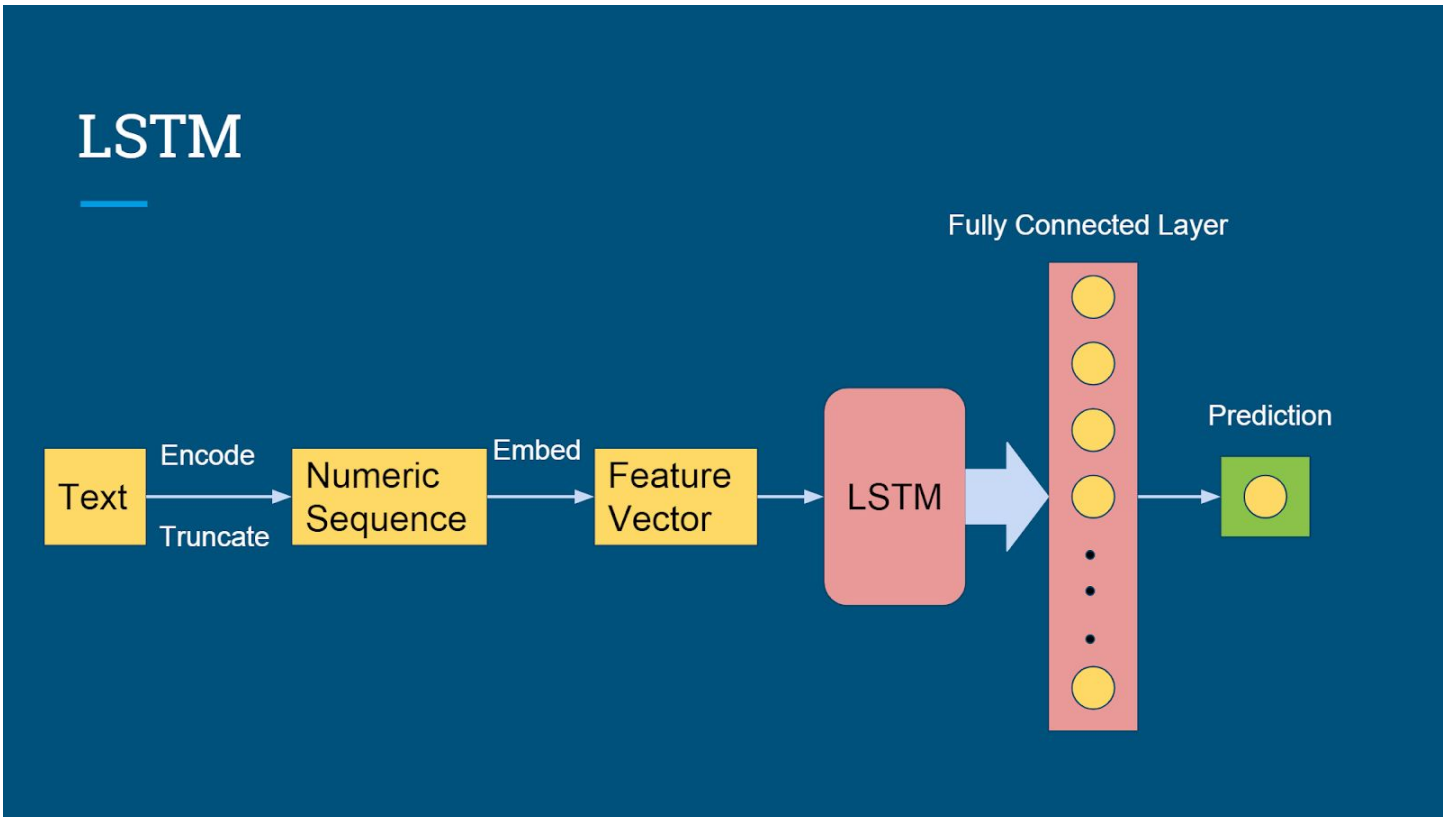
The Long-Short Term Memory (LSTM) unit is good at classifying serialized objects because it will selectively memorize the previous input and use that, together with the current input, to make prediction.
The news content (text) in our problem is inherently serialized. The order of the words carries the important information of the sentence. So the LSTM model suits for our problem. Since the order of the words is important for the LSTM unit, we cannot use the Doc2Vec for preprocessing because it will transfer the entire document into one vector and lose the order information. To prevent that, we use the word embedding instead.

The steps to carry out this process are:

1. Text data is first cleaned by removing all characters which are not letters or numbers.
2. Then, the frequency of each word that appeared in the training dataset is counted to find 5000 most common words and give each one an unique integer ID.
3. Once this is done, we replace each common word with its assigned ID and delete all uncommon words.
4. Since the LSTM unit requires a fixed input vector length, we truncate the list longer than 500 numbers because more than half of the news is longer than 500 words.
5. Then, for those list shorter than 500 words, we pad 0's at the beginning of the list. We also delete the data with only a few words since they don't carry enough information for training.
6. By doing so, the original text string is transferred to a fixed length integer vector while preserving the words order information.
7. Finally, word embedding is used to transfer each word ID to a 32-dimension vector. The word embedding will train each word vector based on word similarity.
   (If two words frequently appear together in the text, they are thought to be more similar and the distance of their corresponding vectors is small.)
8. The pre-processing transfers each news in raw text into a fixed size matrix. This processed training data is fed into the LSTM unit to train the model.

The LSTM is still a neural network. But different from the fully connected neural network, it has cycle in the neuron connections. So the previous state (or memory) of the LSTM unit c_t will play a role in new prediction h_t.

$$
\begin{aligned}
h_t &= o_t \cdot \tanh(c_t) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c_t} \\
\tilde{c_t} &= \tanh(x_t W_c + h_{t-1} U_c + b_c) \\
o_t &= \sigma(x_t W_o + h_{t-1} U_o + b_o) \\
i_t &= \sigma(x_t W_i + h_{t-1} U_i + b_i) \\
f_t &= \sigma(x_t W_f + h_{t-1} U_f + b_f)
\end{aligned}
$$



CALCULATIONS

The best model is chosen with the help of the confusion matrix. Using this, we can calculate precision, recall, and F1 scores.

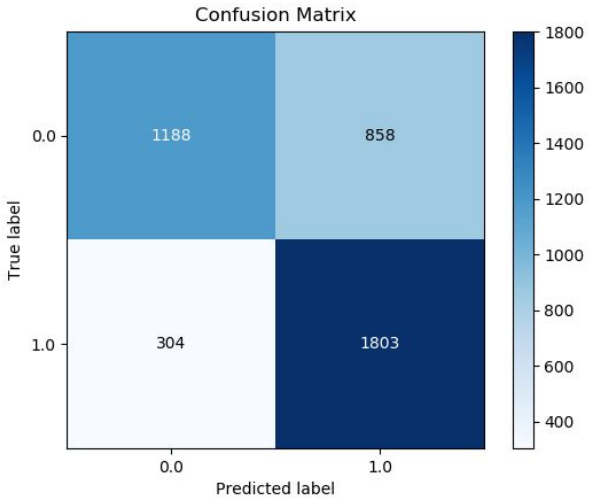
Confusion Matrix

This is a great visual way to depict the predictions as four categories:

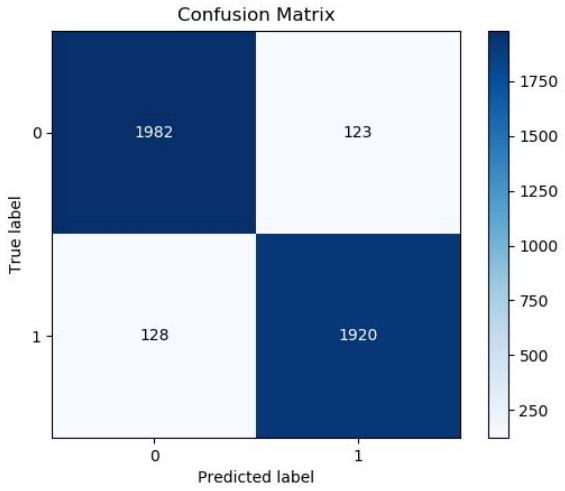
1. **False Positive**: Predicted as fake news but are actually true news.
2. **False Negative**: Predicted as true news but are actually fake news.
3. **True Positive**: Predicted as fake news and are actually fake news.
4. **True Negative**: Predicted as true news and are actually true news

The confusion matrices for each of the models are shown.

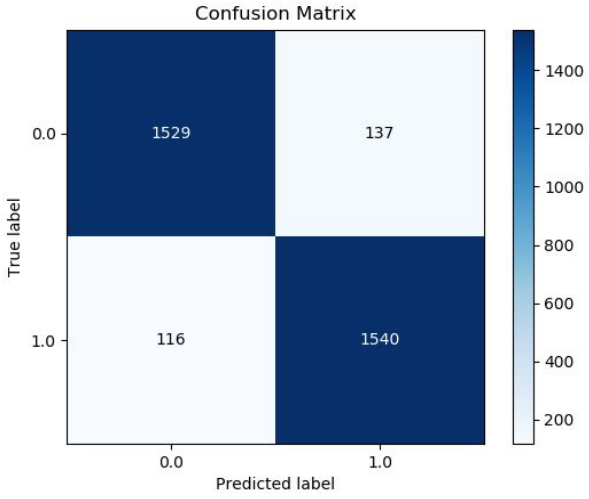
LSTM Confusion Matrix



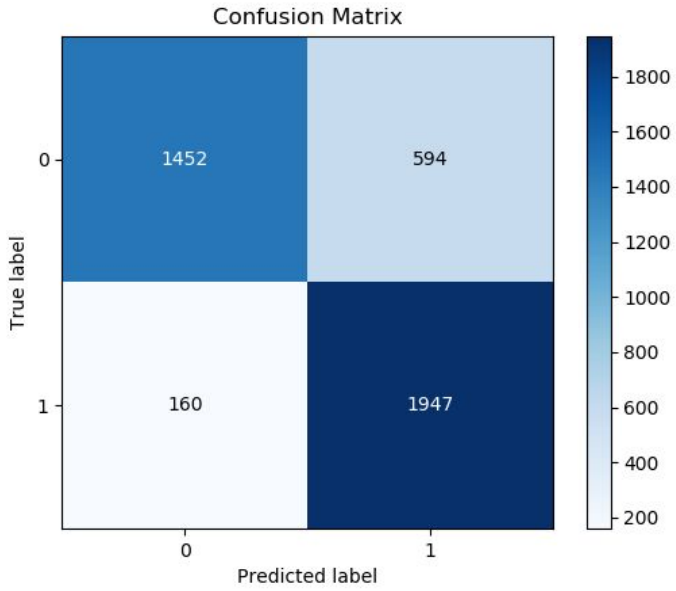
NAIVE BAYES CLASSIFIER



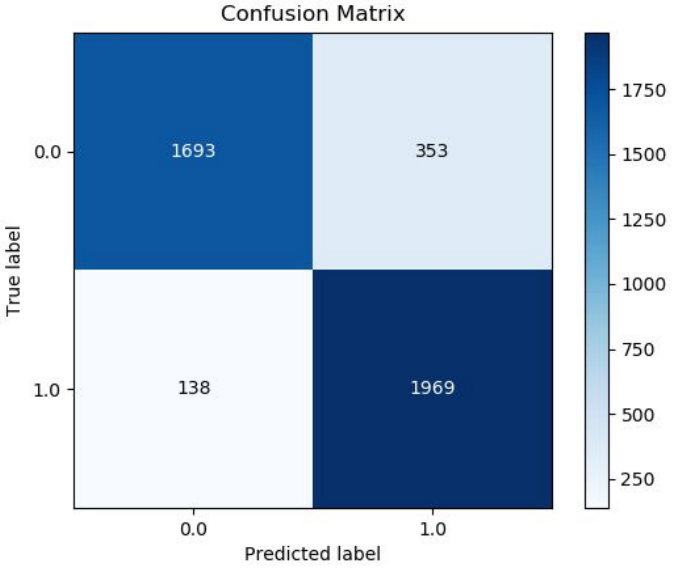
NEURAL NETWORK - KERAS



NEURAL NETWORK - TENSOR FLOW



SVM



Precision and Recall

Precision which is also known as the positive predictive value is the ratio of relevant instances to the retrieved instances.
Precision = No. of True Positives / (No. of True Positives + No. of False Positives)

Recall which is also known as sensitivity is the proportion of relevant instances retrieved among the total number of relevant instances.
Recall = No. of True Positives / (No. of True Positives + No. of False Negatives)
The comparison between the precision, recall and F1 scores for each of the models are tabulated below:

NAME	PRECISION	RECALL	F1
Naive Bayes	0.68	0.86	0.76
SVM	0.85	0.93	0.89
Neural Network with TensorFlow	0.77	0.92	0.84
Neural Network with Keras	0.92	0.93	0.92
LSTM	0.94	0.94	0.94

CHALLENGES FACED

Despite completing the project with a fair accuracy, we faced a few challenges through its course. Some of them include:

- Cleaning the dataset before putting it to use, slowing down our progress.
- Classification based on just content, while there are many other aspects to cover, including classification based on header, footer, etc.
- Not including techniques that help distinguish between click-bait and actual fake news

CONCLUSION

It is evident from the table that LSTM gave us the best results. As mentioned previously, we had to use a different set of embeddings for preprocessing the data to be fed to our LSTM model. It uses ordered set of Word2Vec representations
One of the reasons that LSTM performs so well is because the text is inherently a serialized object. All the other models use the Doc2Vec to get their feature vectors and hence, they rely on the Doc2Vec to extract the order information and perform a classification on it. On the other hand, the LSTM model preserves the order using a different preprocessing method and makes prediction based on both words and order, thus outperforming all other models.

ACKNOWLEDGEMENTS

We sincerely thank Dr. Rajeswari Sridhar for giving us the opportunity to work on this insightful project and guide us through the course of it.

REFERENCES

[1] Datasets, Kaggle, https://www.kaggle.com/c/fake-news/data, February, 2018.

[2] Fake news identification by Sohan Mone, Devyani Choudhary, Ayush Singhania
http://cs229.stanford.edu/proj2017/final-reports/5244348.pdf

[3] Some-like-it-hoax
https://github.com/gabll/some-like-it-hoax/tree/master/logistic-regression

[4] Fake-news-detection
https://github.com/ritikavnair/Fake-News-Detection/blob/master/FakeNewsDetectionProjectReport.pdf