FAKE NEWS DETECTION

[1]Harun M. Tamboli, [2]Sayali S. Bhosale, [3]Shraddha S. Dhasade & [4]Shraddha S. Gaikwad

[1]hmtamboli@coe.sveri.ac.in , [2]sayalibhosale909@gmail.com, [3]shraddhadhasade7@gmail.com, [4]shraddhagaikwad1998@gmail.com

[1]Asst. Prof., Computer Science & Engineering,
SVERI's COE Pandharpur, Maharashtra-India.

[2,3,4] UG Students, Computer Science & Engineering,
SVERI's COE, Pandharpur, Maharashtra-India.

Abstract: *In this paper, we explore the application of Natural Language Processing techniques and Machine Learning Algorithms to identify when a news source may be producing fake news. We use a corpus of 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 a text classification approach, using four different classification models, and analyze the results. 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 our article misclassification tolerance, because we then have multiple data points coming from each source.*

Keywords: **Natural Language Processing, Machine Learning, Classification,Fake news Sources**

1. INTRODUCTION

Fake news, defined as a made-up story with an intention to deceive, has been widely cited as a contributing factor to the outcome of the 2016 United States presidential election. While Mark Zuckerberg, Facebook's CEO, made a public statement denying that Facebook had an effect on the outcome of the election, Facebook and other online media outlets have begun to develop strategies for identifying fake news and mitigating its spread. Zuckerberg admitted identifying fake news is difficult, writing, "This is an area where I believe we must proceed very carefully though. Identifying the truth is complicated." 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.

With this paper, we seek to understand and respond to neural fake news before it manifests at scale. We draw on the field of computer security, which relies on *threat modeling*: analyzing the space of potential threats and vulnerabilities in a system to develop robust defenses. To scientifically study the risks of neural disinformation, we present a new generative model.Our model allows for controllable yet efficient generation of an entire news article – not just the body, but also the title, news source, publication date, and author list. Accordingly, we suggest a provisional policy of how such models should be released and why we believe it to be safe – and perhaps even imperative – to do so. We believe our proposed framework and accompanying models
provide a concrete initial proposal for an evolving conversation about ML-based disinformation threats and how they can be countered.

2. 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. We had to do quite a bit of preprocessing of the data, as is evident from our source code, in order to train our models.
A full training dataset has the following attributes:

1. id: unique id for a news article
2. title: the title of a news article
3. author: author of the news article
4. text: the text of the article; incomplete in some cases
5. label: a label that marks the article as potentially unreliable

• 1: unreliable
• 0: reliable

3. FEATURE EXTRACTION AND PREPROCESSING

The embeddings used for the majority of our modelling are generated using the Doc2Vec model. The goal is to produce a vector representation of each article. Before applying Doc2Vec, we perform some basic pre-processing of the data. 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. Doc2Vec is a

model developed in 2014 based on the existing Word2Vec model, which generates vector representations for words . Word2Vec 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. Preservation of word order information makes Doc2Vec useful for our application, as we are aiming to detect subtle differences between text documents.

4. Long Short-Term Memory Model

The Long-Short Term Memory (LSTM) unit was proposed by Hochreiter and Schmidhuber. It 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.

We first clean the text data by removing all characters which are not letters nor numbers. Then we count the frequency of each word appeared in our training dataset to find 5000 most common words and give each one an unique integer ID. For example, the most common word will have ID 0, and the second most common one will have 1, etc. After that we replace each common word with its assigned ID and delete all uncommon words. Notice that the 5000 most common words cover the most of the text, as shown in Figure 1, so we only lose little information but transfer the string to a list of integers. 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 as shown in Figure 2. 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. By doing this, we transfer the original text string to a fixed length integer vector while preserving the words order information. Finally we use word embedding 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.

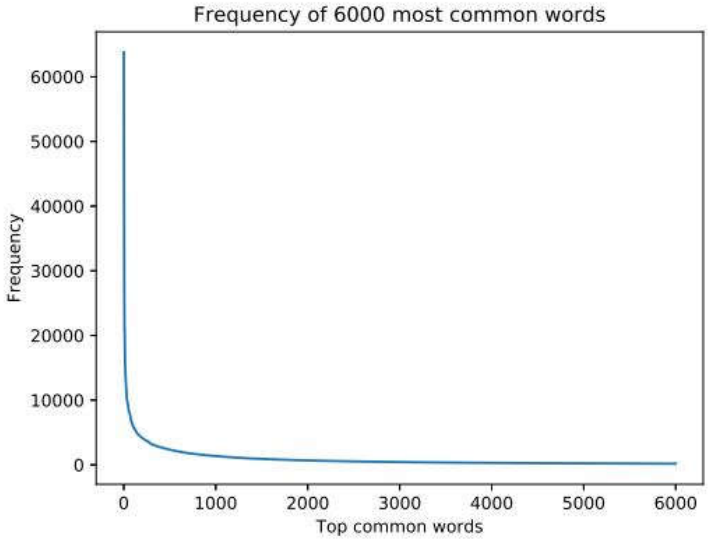

Figure 1 : Frequency of Top Common Words

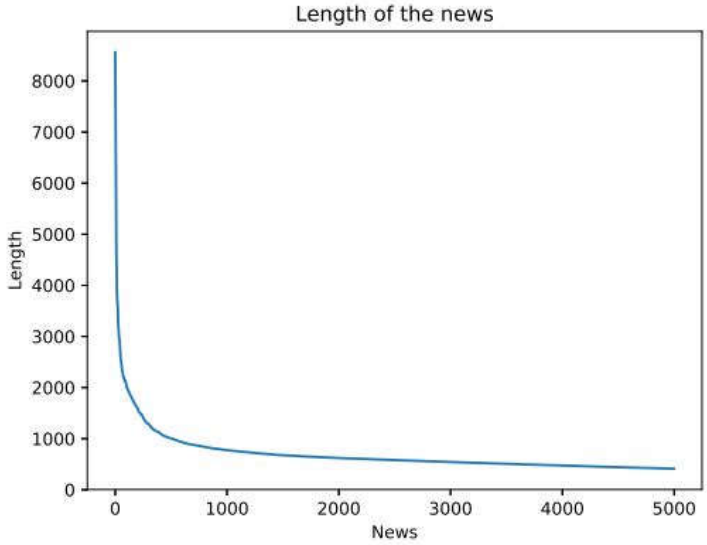

Figure 2: Length of the News

The pre-processing transfers each news in raw text into a fixed size matrix. Then we feed the processed training data 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 *ct* will play a role in new prediction *ht*.

$$ht = ot \cdot \tanh(ct) \quad (1)$$

$$ct = ft \cdot ct\ 1 + it \cdot c\tilde{}t \quad (2)$$

$$c\tilde{}t = \tanh(xtWc + ht\ 1Uc + bc) \quad (3)$$

$$ot = \sigma(xtWo + ht\ 1Uo + bo) \quad (4)$$

$$it = \sigma(xtWi + ht\ 1Ui + bi) \quad (5)$$

$$ft = \sigma(xtWf + ht\ 1Uf + bf) \quad (6)$$

The LSTM achieves the highest F1 score in comparison to all the other models, followed by the Neural Network model . 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 pre-processing method and makes prediction based on both the words and their order. This is how it outperforms others.

ACKNOWLEDGMENTS

Many thanks to Prof. H.M.Tamboli sir and Prof. P.G. Gaikwad sir to guide us throughout the course of this project with his immense knowledge in the field of Machine Learning. Also thanking the teaching faculty and Head of the Department for their guidance.

References :

[1] Zuckerberg, M., *Facebook Post*, https://www.facebook.com/zuck/posts/ 10103253901916271, November, 2016.

[2] Allcott, H., and Gentzkow, M., *Social Media and Fake News in the 2016 Election*, https://web.stanford.edu/œgentzkow/research/fakenews.pdf, January, 2017.

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

[4] Quoc, L., Mikolov, T., *Distributed Representations of Sentences and Documents*, https://arxiv. org/abs/1405.4053, May, 2014.

[5] Christopher, M. Bishop, *Pattern Recognition and Machine Learning*, http://users.isr.ist.utl.pt/˜wurmd/Livros/school/Bishop%20%20Pattern%20Recognition%20And% 20Machine%20Learning%20-%20Springer%20%202006.pdf, April, 2016.

[6] Goldberg, Y., *A Primer on Neural Network Models for Natural Language Processing*, https://arxiv. org/pdf/1510.00726.pdf, October, 2015.

[7] Hochreiter, S., Jrgen, S., *Long short-term memory*. http://www.bioinf.jku.at/ publications/older/2604.pdf, October, 1997