Advanced Machine Learning (MIS 64061)

Dr. Murli Shankar

2021

# Fake News Detection: A Deep Learning Approach

Submitted By:

Rakhee Moolchandani

## Table of Contents

## Abstract

Fake news is defined as a made-up story with an intention to deceive or to mislead. In this report, I present the solution to the task of fake news detection by using Deep Learning architectures. Gartner research predicts that “By 2022, most people in mature economies will consume more false information than true information”. The exponential increase in production and distribution of inaccurate news presents an immediate need for automatically tagging and detecting such twisted news articles. The objective of this project is to build a classifier that can predict whether a piece of news is fake based only on its title, thereby approaching the problem from purely NLP perspective. An important part of the goal is to compare and report the results from multiple different model implementations and present an analysis on the findings.

## Introduction

"Fake News" is a term used to represent fabricated news or propaganda comprising misinformation communicated through traditional media channels like print, and television as well as non-traditional media channels like social media. The general motive to spread such news is to mislead the readers, damage reputation of any entity, or to gain from sensationalism. It is seen as one of the greatest threats to democracy, free debate, and the Western order.

Fake news is increasingly being shared via social media platforms like Twitter and Facebook. These platforms offer a setting for the general population to share their opinions and views in a raw and un-edited fashion. Research that studied the velocity of fake news concluded that tweets containing false information reach people on Twitter six times faster than truthful tweets.

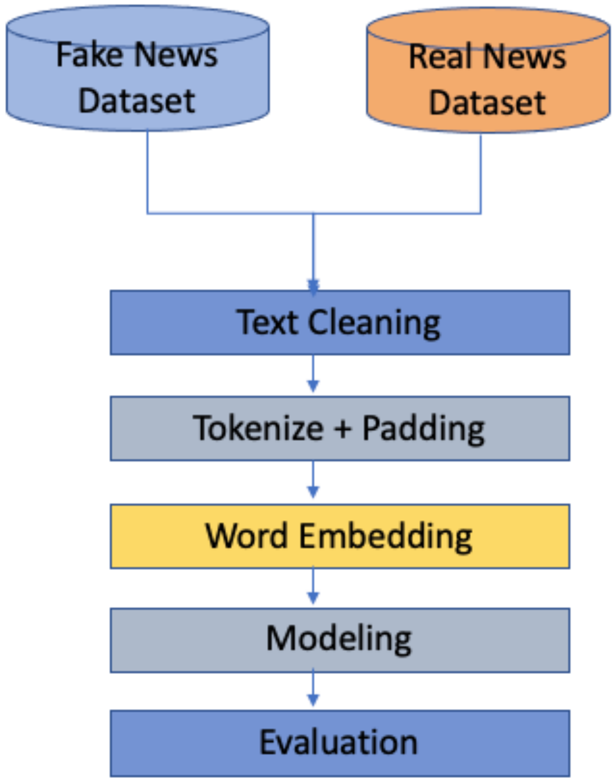
Technologies such as Deep Learning, Artificial Intelligence and Natural Language Processing (NLP) tools offer great solution for researchers to build systems which could automatically detect fake news. However, detecting fake news is a challenging task to accomplish as it requires models to summarize the news and compare it to the actual news in order to classify it as fake.

## The Problem

The problem is not only hackers, going into accounts, and sending false information. The bigger problem here is what we call “Fake News”. A fake are those news stories that are false: the story itself is fabricated, with no verifiable facts, sources, or quotes. When someone (or something like a bot) impersonates someone or a reliable source to false spread information, that can also be considered as fake news. In most cases, the people creating this false information have an agenda, that can be political, economic or to change the behavior or thought about a topic.

There are countless sources of fake news nowadays, mostly coming from programmed bots, that can't get tired (they're machines) and continue to spread false information 24/7.
The tweets in the introduction are just basic examples of this problem, but much more serious studies in the past 5 years, have demonstrated big correlations between the spread of false information and elections, the popular opinion or feelings about different topics.
The problem is real and hard to solve because the bots are getting better are tricking us. Is not simple to detect when the information is true or not all the time, so we need better systems that help us understand the patterns of fake news to improve our social media, communication and to prevent confusion in the world.

## Workflow

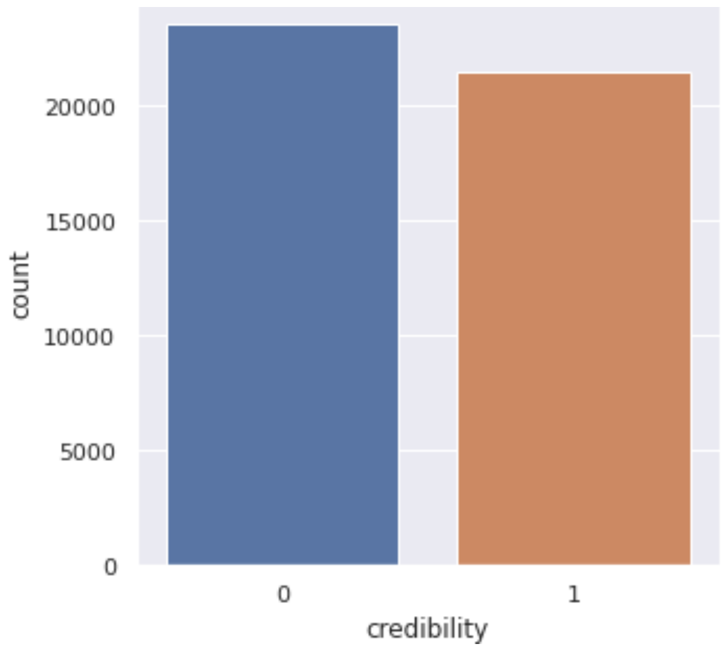


## The Data

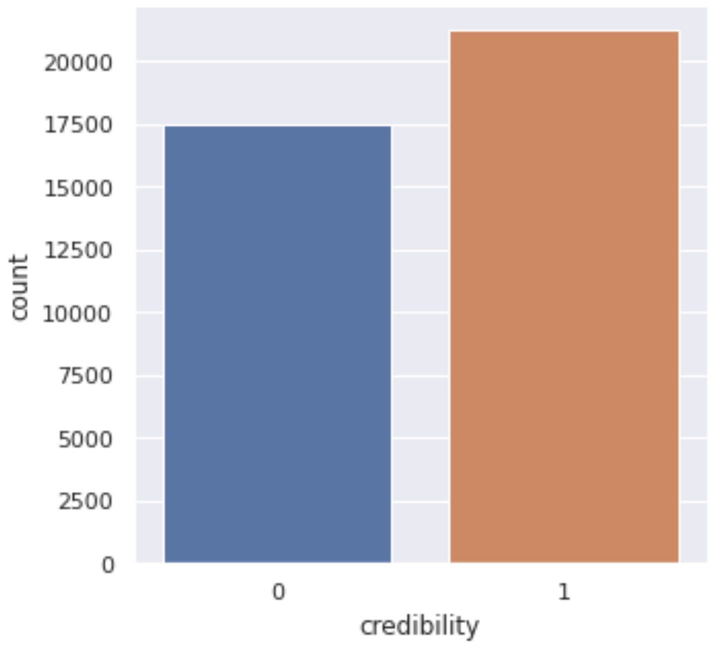
The data comes from Kaggle and Signal Media News datasets. There are two files, one for real news and one for fake news (both in English) with a total of 23481 "fake" and 21417 "real" articles.

## Data Exploration

How many fake and real?

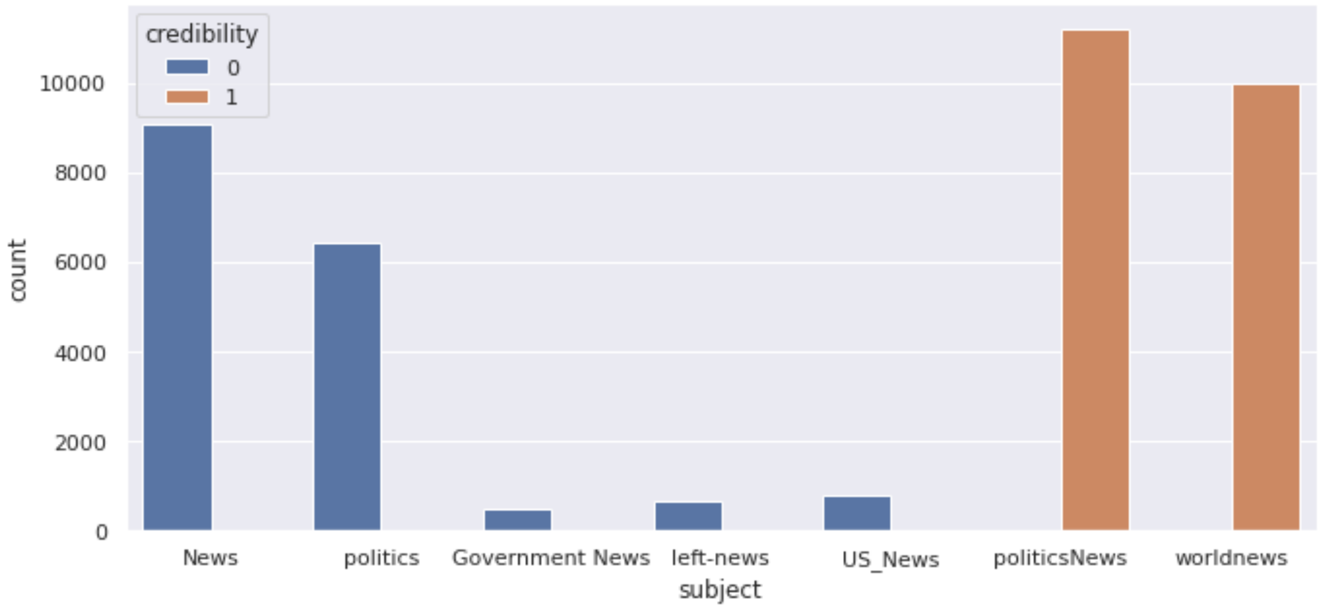


From the above, it is clear that the dataset is balanced for both fake and real news. The 0 credibility states the fake news whereas credibility 1 shows the real news. Although, there were 5140 duplicate texts. We need to remove those duplicate rows.



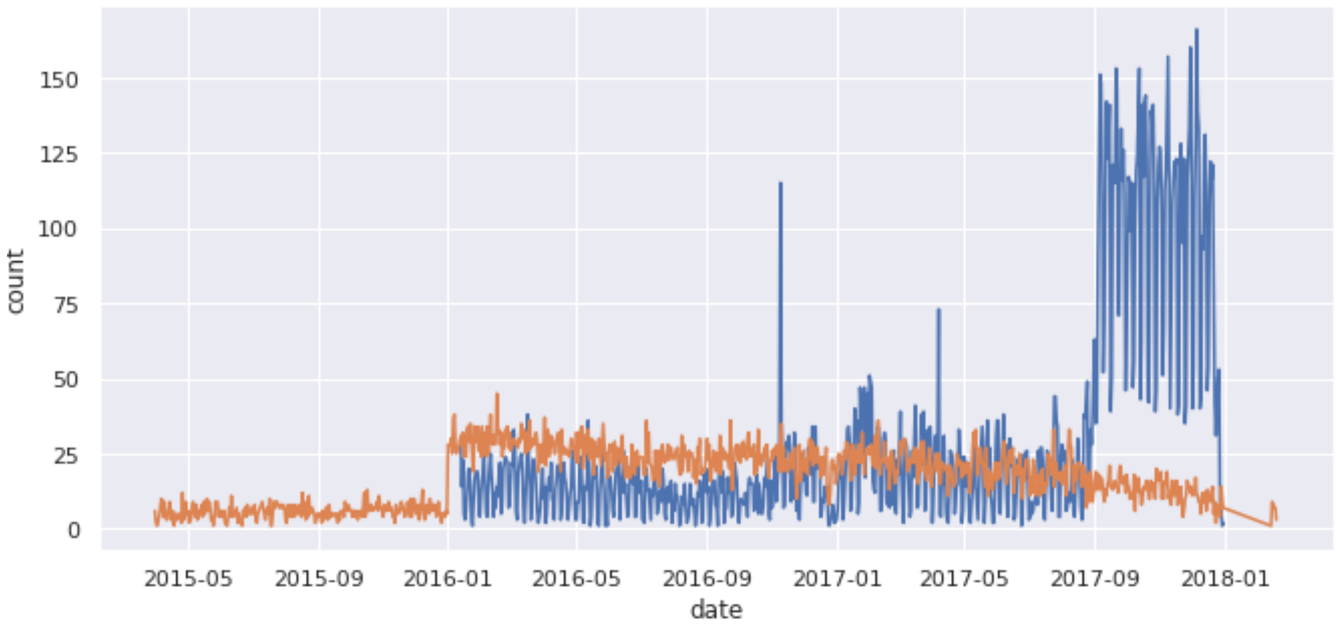
After dropping duplicates, the count of fake news has reduced, meaning most of the duplicate text were from fake news. However, the dataset set is still balanced.

Checking for relationship between news subject and news credibility:



- From the plot above, it is clear that real news is only centered around politic news and world news subject areas, while fake news is centered around the other subject areas.
- This indicates that the subject area can help determine if news is fake or real.

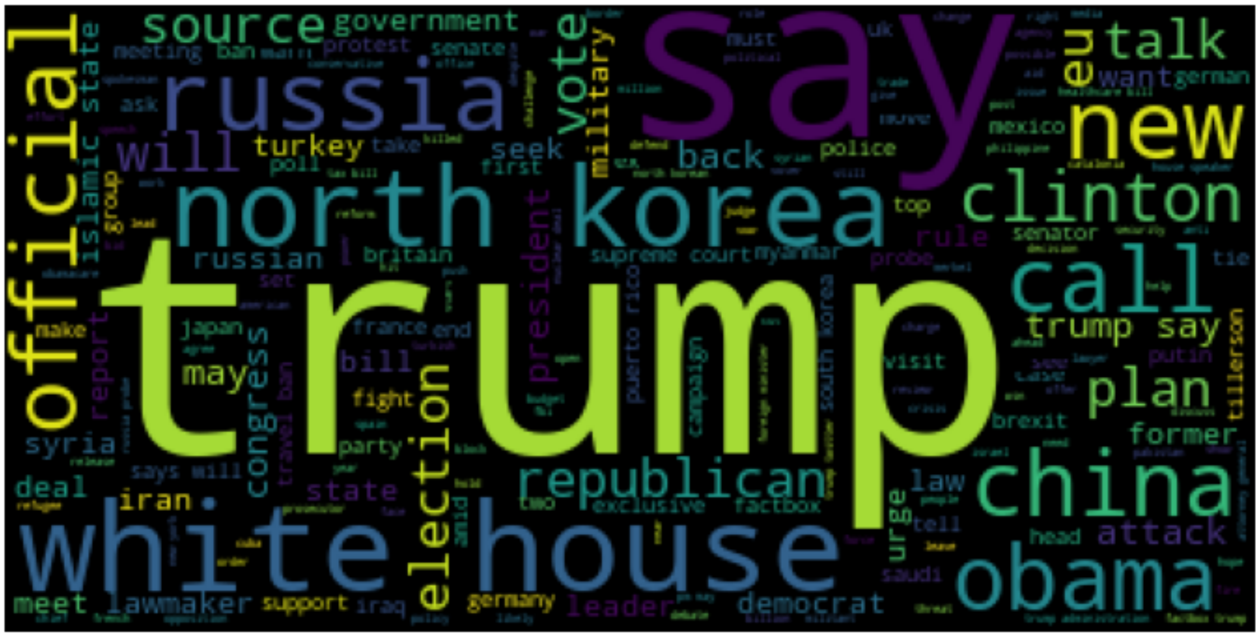
Checking for relationship between news date and news credibility:



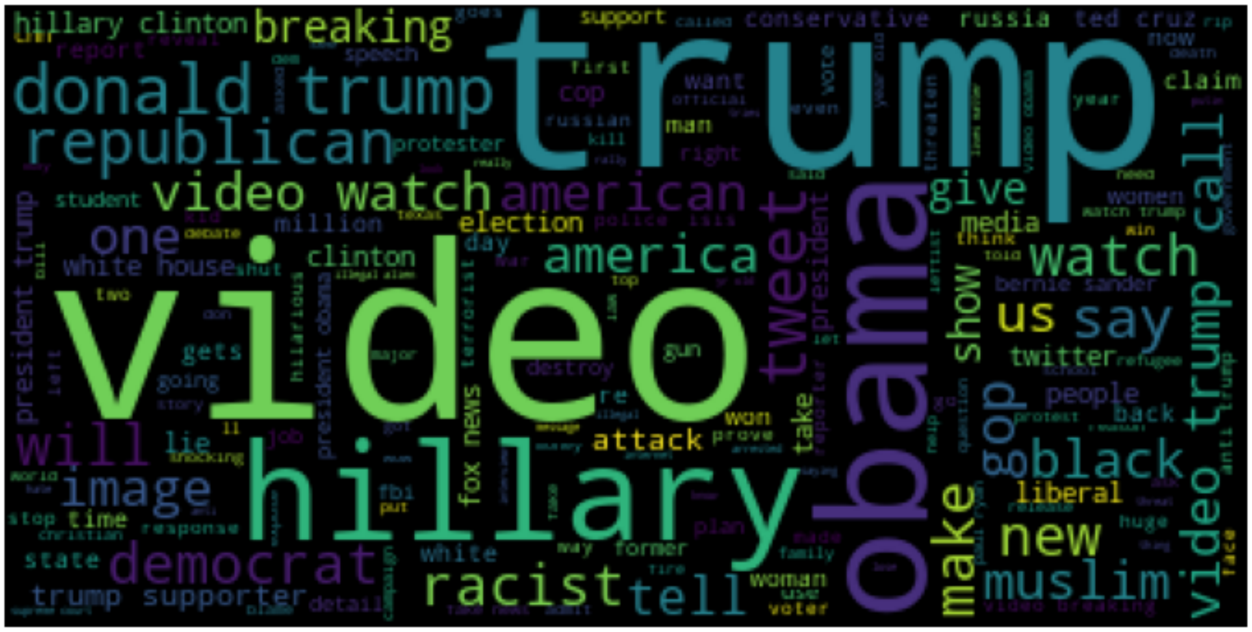
From the plot below, it seems there is some correlation between date a news article was created and its credibility. There was a sharp rise in fake news in later years, while real news dropped marginally.

Checking for relationship between news title, news text and news credibility:

word cloud for real news title



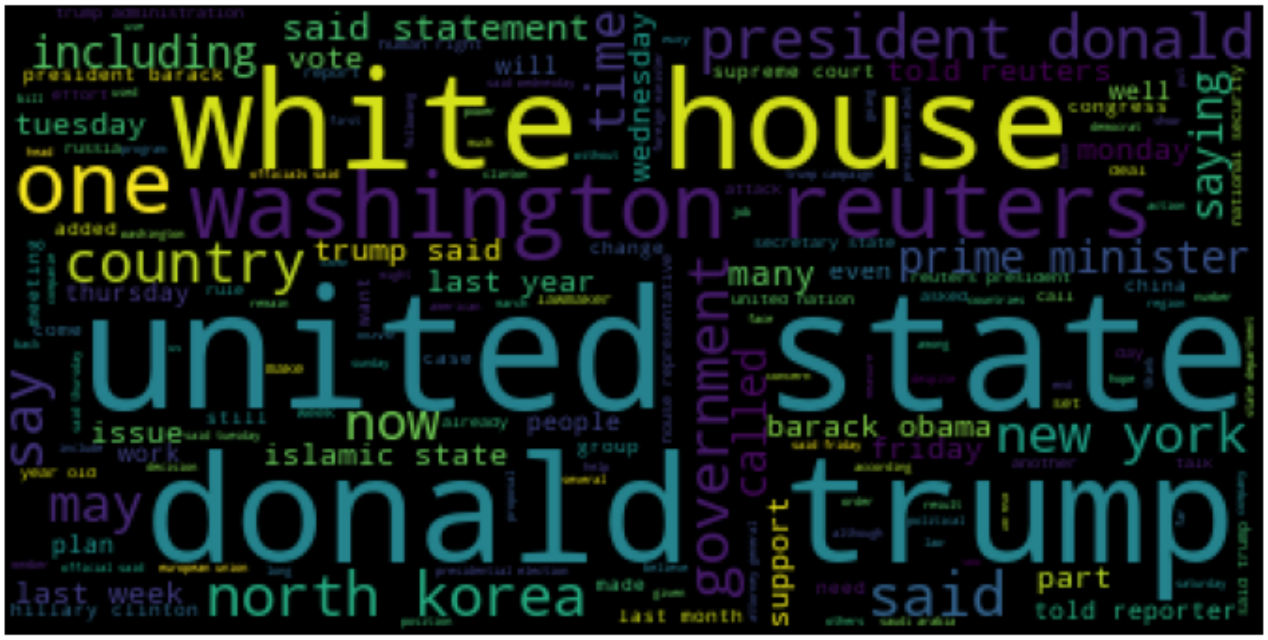
word cloud for fake news title



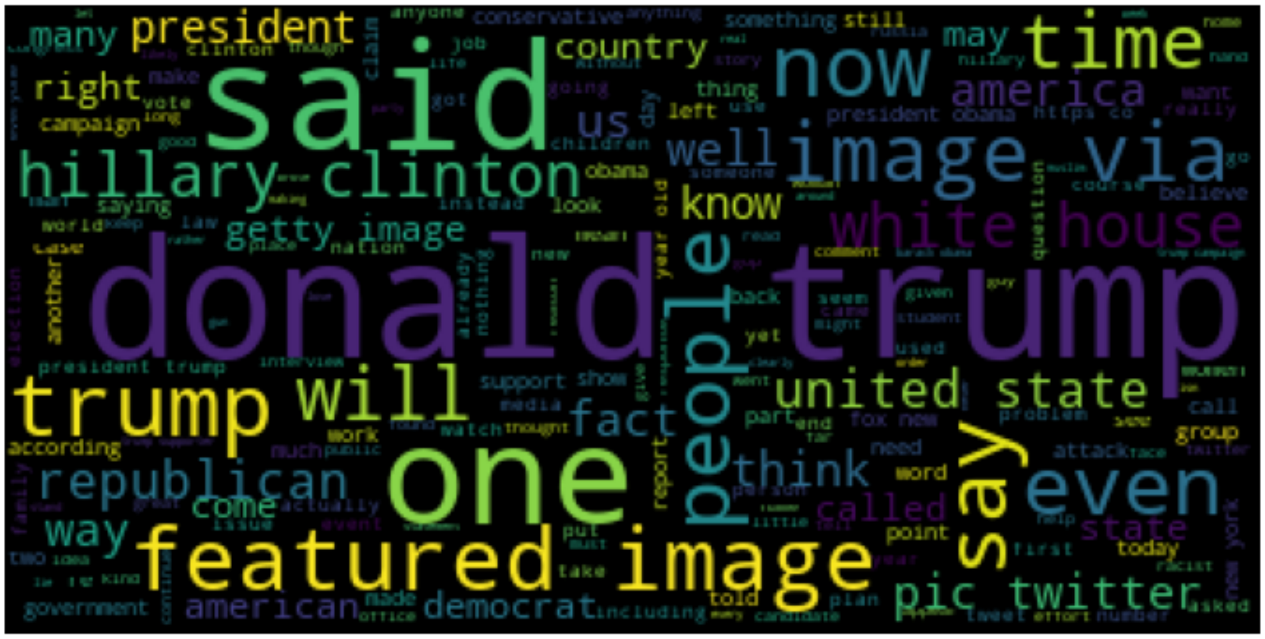
Similar common words in both fake news and real news titles include: Trump, Obama, etc. But there are words like White House, US, North Korea, Russia, that are very common in real news titles but are not so common in fake news titles. On the other hand, there are words like Video, tweet, hillary, watch, gop, that are common in fake news titles, but are not so common in real news titles. This shows that there is some distinguishing feature between most real and fake news titles, and including titles in our analysis can add some information to our model

Checking for relationship between news text and credibility

word cloud for real news text



word cloud for fake news text



Although there are similar common words in both real news text and real news titles, there are still some distinguishing common words like people, featured image, percent, wednesday, thursday, tuesday, US, one, etc. This shows that the text of a news article is also a determinate factor in its credibility.

## Data Preprocessing

Removing punctions and unneeded characters from news text:
In English, a contraction is a word or phrase that has been shortened by dropping one or more letters, such as “I’m” instead of “I am”. We can either split the contractions (“I’m” to “ I ”+” ’m ”) or convert them to their full format (“I’m ” to “I am”). In my experience the latter works better as it's harder to find a word embedding for sub-words like “ ‘m “.We want the sentences without punctuations like commas, brackets, etc. Python has a constant called string.punctuation that provides a list of punctuation characters. We’ll use this list to clean our text from punctuations. I noticed after all this cleaning, there were still some words like “…but” with dots in them. I added this last step to clean them up.

Removing stop words:
Stopwords are common words like “the”, “and” … which don’t add much value to the meaning of the text. NLTK has a list of these words that can be imported and used to remove them from the text.

Tokenization:
In most of the NLP tasks, we need to represent each word in the text with an integer value (index) before feeding it to any model. In this way, the text will be converted to a sequence of integer values. One of the ways of doing that is with Keras. Keras provides an API for tokenizing the text. Tokenizer in Keras finds the frequency of each unique word and sort them based on their frequency. It then assigns an integer value starting from 1 to each word from the top. You can see the index mapping dictionary by reading tokenizer.word_index.

There are two distinct steps in tokenizing in this way:

1. fit_on_texts: We'll fit the tokenizer on our training data to create the word indices
2. texts_to_sequences: using the word index dictionary from step above, we take this step to transform both train and test data.

Padding and truncating the input training sequences:
All input sequences to the model need to have the same length. In order to achieve that, we can use a function that pads the short sequences with zeros (options are 'pre' or 'post' which pads either before or after each sequence). It also truncates any sequence that is longer than a predefined parameter "maxlen". Truncating also has the option of 'pre' or 'post' which either truncates at the beginning or at the end of the sequences.

Word Embedding from scratch:
Now that we have tokenized the text, we developed the code from scratch for work embedding. Word embeddings is a way to represent words with similar meaning to have

a similar representation. Although my models did not perform well and were overfit with this process.

Word embedding using pre-trained GloVe vectors:
Now we use GloVe pretrained vectors.  Using word embedding through GloVe, we can have a decent performance with models with even relatively small label training sets.

Prepare train and test datasets:
I have used the usual train_test_split by sklearn to split the data.

## Feature Extraction and Model Training

Several Different models were implemented, the following section describes each one of them. All models use pre trained 100 dimensional GloVe embeddings.

Recurrent Neural Network with LSTM:
RNN is widely used neural network architecture for NLP. It has proven to be comparatively accurate and efficient for building language models and in tasks of speech recognition. RNNs are particularly useful if the prediction has to be at word-level. Although a RNN can learn dependencies however, it can only learn about recent information. LSTM can help solve this problem as it can understand context along with recent dependency. Hence, LSTM are a special kind of RNN where understanding context can help to be useful. LSTM networks are similar to RNNs with one major difference that hidden layer updates are replaced by memory cells. This makes them better at finding and exposing long range dependencies in data which is imperative for sentence structures.

Recurrent Neural Network with GRU:
Similarly, Gated Recurrent Units are designed in a manner to have more persistent memory thereby making it easier for RNNs to capture long term dependencies.

Convolution Network with Max Pooling:
CNNs are generally used in computer vision, however they've recently been applied to various NLP tasks. CNN is a class of deep, feed-forward artificial neural networks (where connections between nodes do *not* form a cycle) & use a variation of multilayer perceptron designed to require minimal preprocessing.

## Implementation

All the code was run in Python using TensorFlow and NumPy. Dropout was used as a regularization mechanism. All the experiments were run on the local CPU machine.

Hyperparameters for the Different Models

| Model | Embedding Size | Optimizer | Dropout | Epochs |
|---|---|---|---|---|
| LSTM | 100 | Adam | 0.3 | 10 |
| GRU | 100 | Adam | 0.3 | 10 |
| CNN with Max Pooling | 100 | Adam | 0.3 | 10 |
| CNN with global Average Pooling | 100 | Adam | 0.3 | 10 |

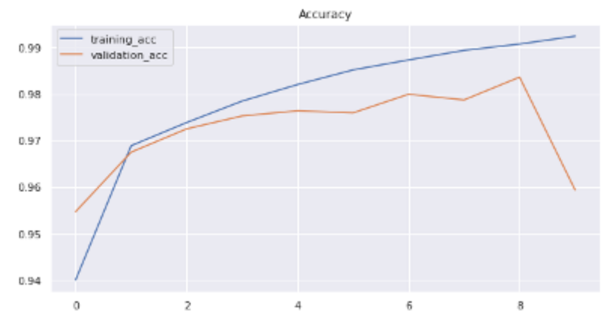
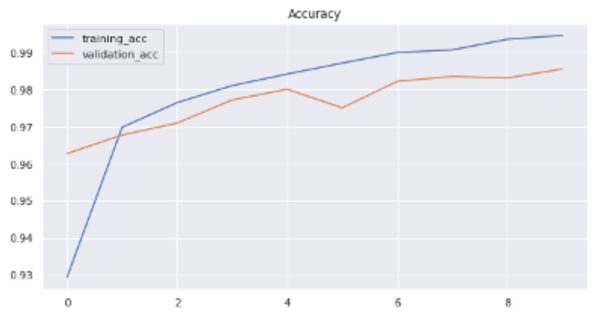
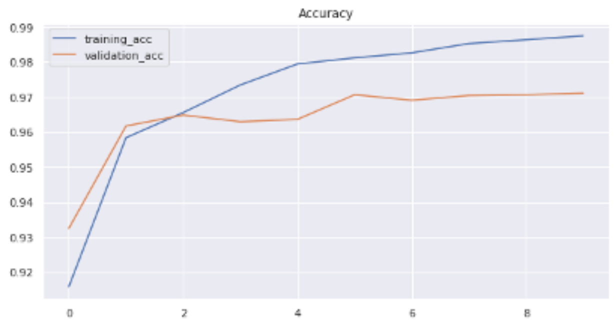
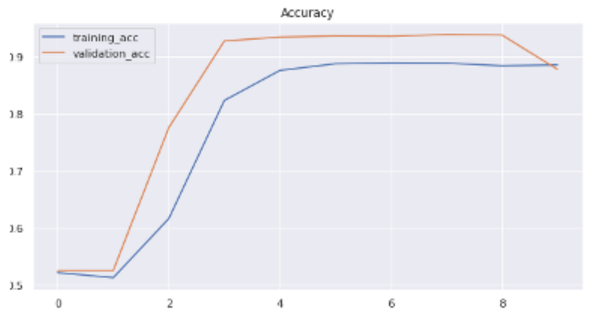
## Code

Along with the report, I am submitting 2 jupyter notebook files.

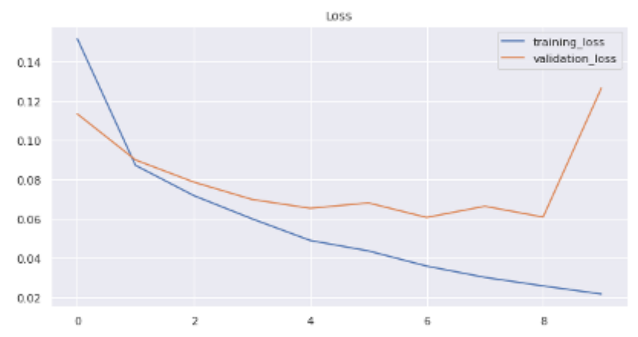
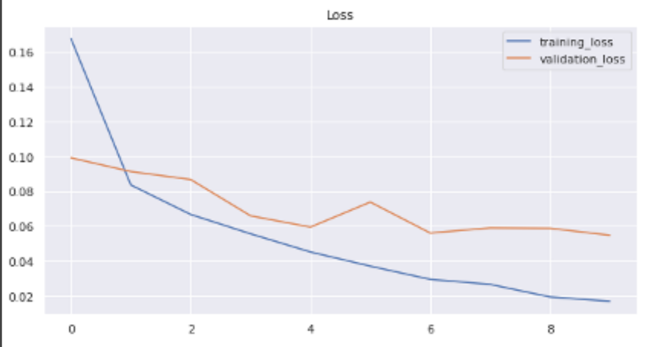
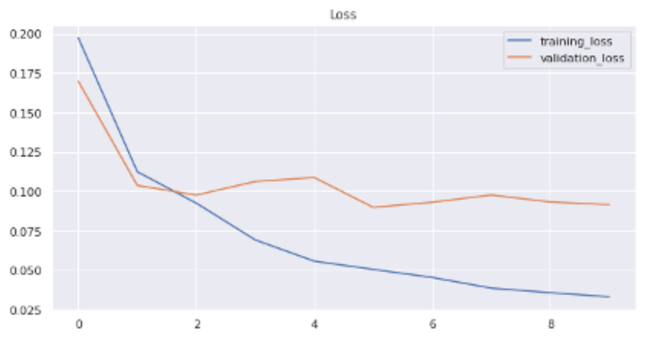
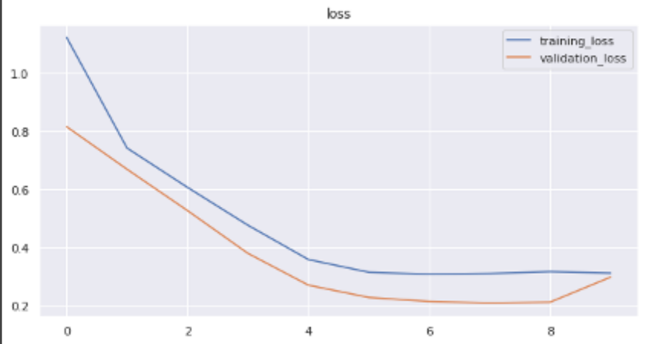
1) AML Final Project: This file includes code for word embedding from scratch as well as glove pretrained network where my models are overfitting. This was my first attempt of code.
2) AML Final Project LSTM & GRU: This file includes Glove pretrained word embedding process along with multiple models. The results of this I have explained below.

## Results

Accuracy of Models:

| LSTM | GRU |
|---|---|
|  |  |
| **Maxpooling** | **Global Average Pooling** |
|  |  |

Loss of Models:

| LSTM | GRU |
|---|---|
|  |  |
| **Maxpooling** | **Global Average Pooling** |
|  |  |

Comparison of all the models:

| Model | Training Accuracy | Test Accuracy | Confusion Matrix |
|---|---|---|---|
| LSTM | 96.93 | 95.94 | `array([[4669, 53], [312,3946]])` |
| GRU | 99.72 | 98.55 | `array([[4641, 81], [ 49,4209]])` |
| CNN with Max Pooling | 99.59 | 97.10 | `array([[4573, 149], [111,4147]])` |

| | | | |
|---|---|---|---|
| CNN with global Average Pooling | 87.13 | 87.77 | `array([[4606, 116],`<br><br>`[982,3276]])` |

## Limitations

However, automated detection of fake news is a hard task to accomplish as it requires the model to understand nuances in natural language. Moreover, majority of the existing fake news detection models treat the problem as a binary classification task, which limits model's ability to understand how related or unrelated the reported news is when compared to the real news.

## Future Directions

For Fake news detection, we can add more features beyond the vectors corresponding to the words in title or text such as source of news, any associated URLs, the topics, publishing medium, country or region of origin etc.
I also believe that the idea of attention enabled CNNs is a promising model worth exploring further. So, I plan to continue this effort after the class ends.

## References

https://towardsdatascience.com/how-to-build-a-recurrent-neural-network-to-detect-fake-news-35953c19cf0b
https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview
https://towardsdatascience.com/detecting-fake-news-with-and-without-code-dd330ed449d9
https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2710385.pdf
https://www.scitepress.org/Papers/2021/103160/103160.pdf
https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f
https://www.youtube.com/watch?v=MXPh_lMRwAI