# CS 410 Project Report: Fake News Detection

Ningyuan Zhang(nz13@illinois.edu),
Jacky Sa(mjsa2@illinois.edu),
Sharanya Balaji(sbalaji3@illinois.edu)

## 1 Project Background

With the availability of news exploding across social media, it is very important to filter out fake news to prevent their damage on society. We believe that deep learning techniques can leverage the sequence of words to model the likelihood of a news being fake. And therefore we choose Fake news detection as the topic of our project.

The goal of the project is to develop neural models, such as RNN model and various model improvements such as BERT to predict the fakeness of a news article.

## 2 Dataset and Approaches

The dataset we chose for this project is Fake News labeled dataset from kaggle: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset, which you can also find in our repo. This dataset has redundant content which reflects real world news as long as reasonable fake news which makes it a good candidate for our project.

Our approach includes the following blocks: 1. EDA, 2. text preprocessing, 3. model architecture search and parameter tuning, 4. result comparison and 5. conclusion. The main language used is Python and main tools used are Pytorch and Transformers by Huggngface.

## 3 Exploratory data analysis (EDA) and Data Preprocessing

The concept of an exploratory data analysis is to analyze and pre-processing the data in order to have the data cleaned to be able to summarize the main characteristics of the data. For this project, we start with two datasets:

1. True News Dataset where all the data is considered real. The columns include title, text, subject, and date.
2. Fake News Dataset where all the data is considered fake. The columns include title, text, subject, and date.

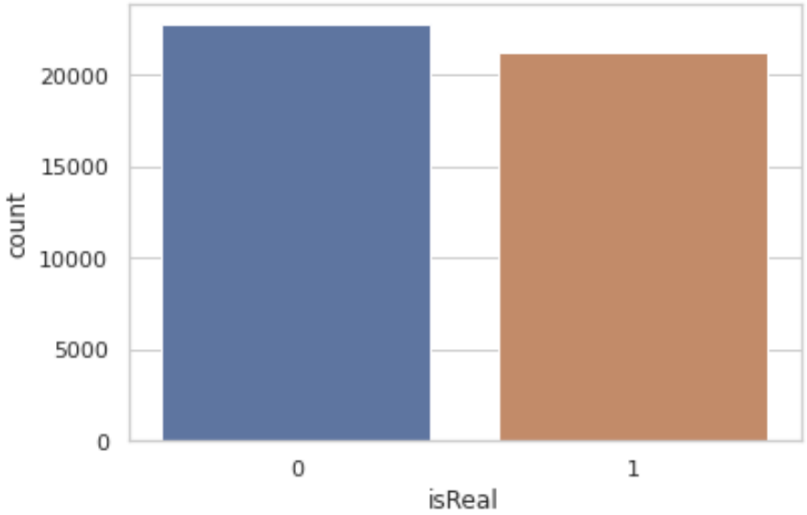
In this specific exploratory data analysis, we first start by adding a column titled “isReal” to both of the datasets. This column will indicate whether the news is considered real or fake. All the data in the true news dataset will have the value of isReal be set to 1 and the data in the fake news dataset will have the value of isReal be set to 0. We then combine these two datasets into one large dataset title news that includes both real and fake news data.

### 3.1 Remove Duplicates

We then remove any duplicate rows in this dataset to ensure that any duplicates do not misrepresent the data in any way. During this process, we delete 209 duplicate records. We also want to remove any null values in the dataset; however, no null values are detected.

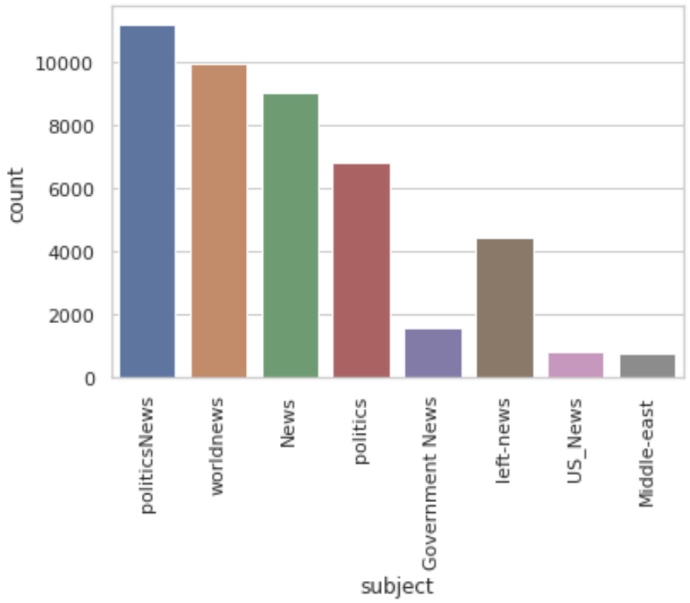
### 3.2 News Label Distribution

At this point, we see a display between real and fake data and we do see that we have more fake data than real data but the overall distribution is balanced, which means accuracy could be a good metric candidate for our problem.




### 3.3 News Subject Distribution

We also display a bar chart that summarises the number of data values that are associated with certain subjects of news. From this, we can notice that the main subject of this news dataset is politics related.




### 3.4 Text Preprocessing

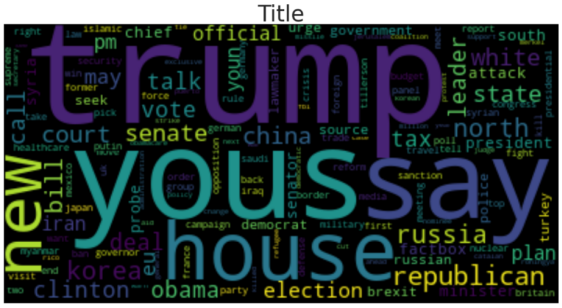
The next step in this exploratory data analysis is to clean the data in order to extract the correct information. The data cleaning steps taken are as follows:

1. Expand all contractions such as don't → do not, isn't → is not
2. Change all text to lowercase. This means that all words will have only lowercase letters.
3. Remove any whitespace between words.
4. Remove any punctuation marks.
5. Removing any numbers.
6. Removing any stop words from the data.

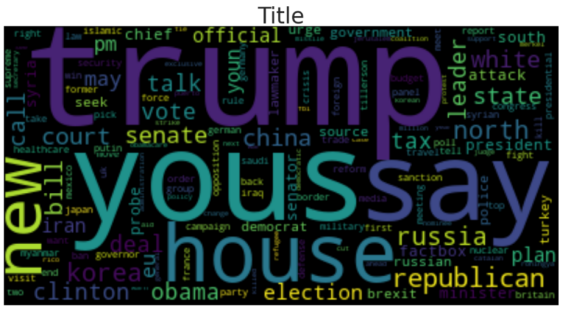
### 3.5 Keyword Analysis: WordCloud for Real News v.s. Fake News

We summarize the main characteristics of the data by displaying word clouds for both the real and fake news. From these word clouds, we can see the most common words in these two data sets.

Real news:



Fake news



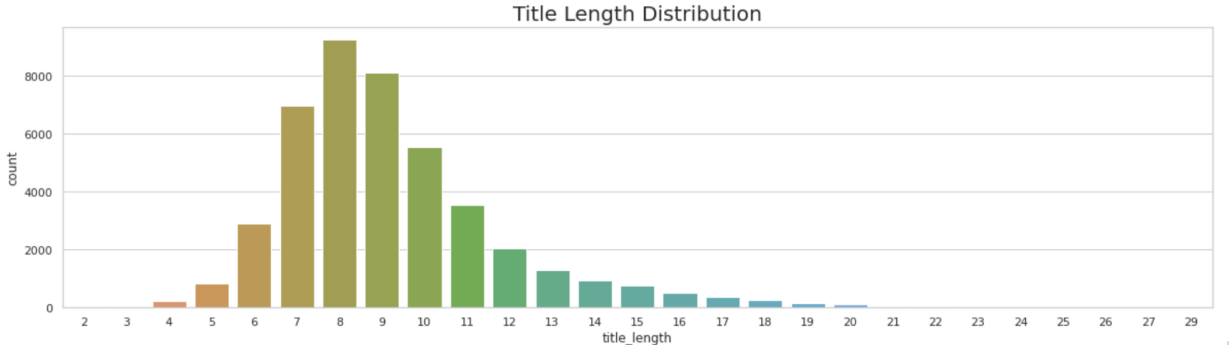

### 3.6 Top N-gram Analysis

We include a n-gram analysis for real news and fake news to look into details on how top words/phrases are different in two distributions. We also conducted the n-gram analysis for each subject to understand the top words for each subject. The finding is not too informative as Trump and president election related words came top in all subjects. Therefore, we won't list pictures here. Please see the EDA code for details.

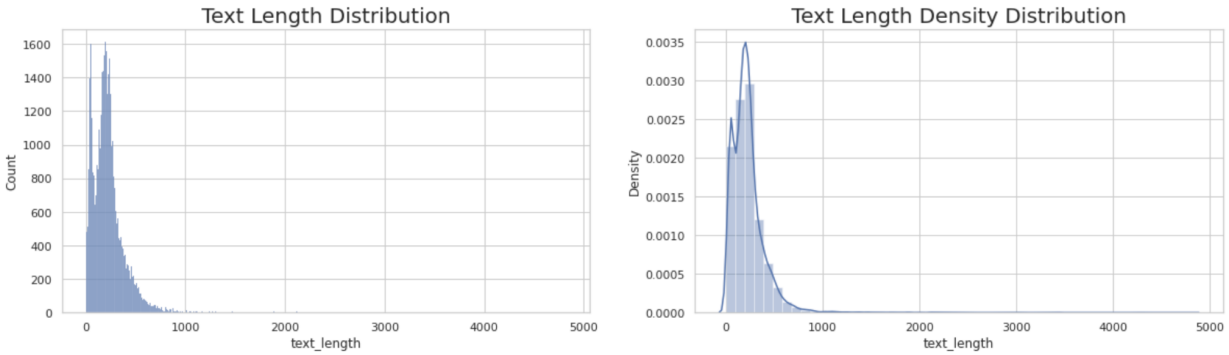
### 3.7 New Length Analysis

We also did some analysis associated with the length of the news. First we filter out news with null content in title or text. Then we also found out that when text length is 1, the text content is not the real new content but the homepage or link address. After filtering out all invalid news, we draw the distribution for text length and title length.

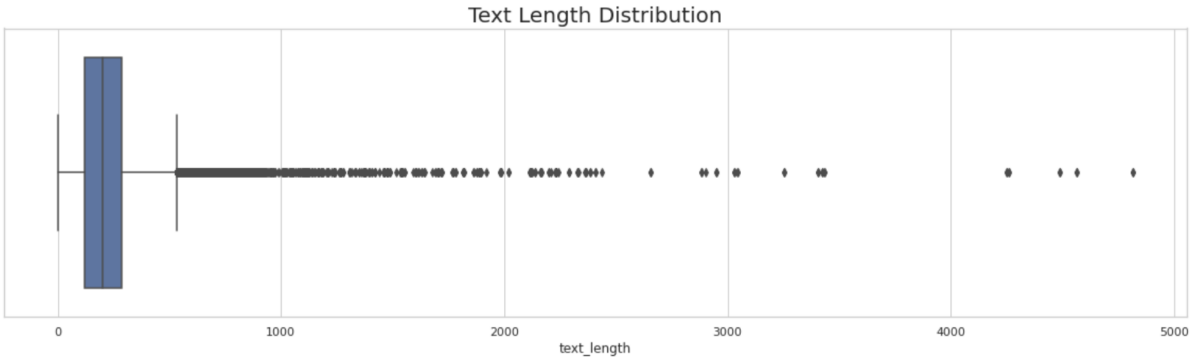
Title length distribution: looks normal, most title length lies in 6-12 words.




Text length distribution: As shown in the plots below, there are some news with extremely long text, but the majority distribute around 500.




Draw a box plot to have a more quantitative understanding of the text length distribution.




We then calculate the percentiles of this boxplot: 25% percentile of text length is 121.0, 50% percentile of text length is 203.0, and 75% percentile of text length is 286.0.

This result is important because later for word embedding, we will need to choose the max_length to prune and pad the news corpus. According to the title and text length distributions, we can conclude that when combining title and text, the length of each news would majorly lay under 300. Therefore, we suggested choosing 300 as max token length for padding and pruning, which can contain most of the news information and discard unnecessary computation and memory waste for long news.

### 3.8 Train/Test Dataset Split

After processing and analyzing the data in its preliminary form, we generate a training and test dataset at a ratio of 80% : 20% so that we can guarantee our models consume consistent dataset for training and evaluation.

## 4 Model Architecture

### 4.1 Embedding

In order to represent words uniquely in a lower dimensional space, we have to “embed” the words as vectors. We chose to Glove pre-trained embedding, since it leverages word semantics and is trained on a large Wikipedia corpus.

### 4.1 Baseline Model: RNN

The RNN, or Recurrent Neural Net model uses hidden states from the word sequence to make predictions. This is a bi-directional model such that the word sequence is read forward and backward. As a result, feeds the hidden state information, or surrounding word sequences into the neural network to accurately make predictions.

The RNN architecture uses TorchText to automatically load the training data as tables, and create the fields for the words and y_labels (fake or real news). The training data is also split into validation data and training data. The validation data is used later to evaluate the loss in the network, so that the weights and bias in the network nodes can be updated.

The text data is loaded into the network in batches of similar size using an iterator. The text tensors in the batches are padded with 0 tokens, to ensure they are the same length before passing through the neural network.

Ultimately, these batches of training and validation data are passed through the network multiple times, and a loss function is used to optimize the weights/bias of the neural network nodes to make predictions.

The result of the RNN model is quite good, and is able to reach 99.5% accuracy on the test dataset. However, we wonder if there can even be stronger improvements and we explored BERT architecture in the next section.

### 4.2 BERT

The BERT model, short for Bidirectional Encoder Representations for Transformer, is the current state of art model for language representation for both token and sentence level tasks. In our project, beyond basic RNN based models, we believe BERT would be a great candidate to investigate, as its architecture is purely based on attention mechanisms and has no recurrence or convolution structure.

We leveraged the Transformers package from HuggingFace to finish the implementation of BERT. Transformers package offers many pre-trained BERT models for different downstream tasks. For fake news detections, we choose to use the BertForSequenceClassification model series, with the basic bert tokenizer and the pre-trained uncased 12-layer BERT Classification model as initial start. By downloading the pre-trained model, we already have a well designed simple BERT with weights pre-trained on a large corpus of general text. Next, we added another

simple feedforward NN layer on top of that and did transfer learning on our news dataset and fine tuned the model's initial weight toward our tasks.

The result of BERT is fairly good, with only 10000 training records, it can achieve 100% accuracy on validation/test dataset at the first epoch (To reduce memory consumption, we first tried with only 10000 records for training). And after 3 epochs, the loss of both training and validation set has dramatically reduced, with the validation accuracy remaining at 100%. Given the results, we just fixed the parameters (learning rate is the main parameter to tune) we used as there's no need for further parameter tuning with the perfect accuracy score. Further tuning could result in overfitting instead of better optimization.


## 5 Conclusion

The Neural Network models overall perform fairly well in our Fake News Detection Task, especially BERT, which has been trained in large corpus of texts and has very powerful capability for language representation and can easily learn the different distribution/representation of real news and fake news. Given the performance, the models should be mature enough to be applied in a real application for fake news detection.

## 6 Using the Code/Instructions

There are three main scripts for this project, one for EDA and data processing, one for RNN/LSTM and one for the BERT model. The scripts are implemented in Google Colab.

1. Download the dataset and scripts folder from github;

2. Upload the dataset and scripts to your google drive directly under “My Drive” (please upload the folder as a whole otherwise you might need to change the path to read files inside the scripts). After this step, we should have “My Drive/CS410Project/...”;

3. Re-run the scripts that you are interested in and tune the parameter to create a model.

For the RNN model, you can train the data by running the code and changing model parameters. Do not alter the way the data is read in through torchtext. However, feel free to change the parameters such as vocab size, epochs.

For the BERT model, we highly recommend that you choose to run on GPU instead of CPU as it will take forever to train BERT with CPU. And please only train the model with a sample of the training data, such as a 10k news sample set, otherwise you will get out of memory issues from Colab.

Please watch the demo for detailed instructions.

## 7 Teamwork

| Name | Work | Hours |
|---|---|---|
| Ningyuan Zhang *Captain | 1. EDA-text length analysis;<br>2. Model- BERT;<br>3. Documentation: report & code instruction;<br>4. Demo;<br>5. Meetings and discussion;<br>6. Github management | 30+ |
| Jacky Sa | 1. Model- RNN;<br>2. Documentation: report & code instruction;<br>3. Demo;<br>4. Meetings and discussion | 30+ |
| Sharanya Balaji | 1. EDA;<br>2. Documentation: report(partial EDA)<br>3. Meetings and discussion | ~20 |