SEP 788/789 – Deep Learning and Neural Networks

Fake News Detection

Instructor: Hamidreza Mahyar

TA: Amir Umani

Name	Student Number	Email
Ruiqiao Wang	400353790	wangr228@mcmaster.ca
Siqi Zhao	400397742	Zhaos98@mcmaster.ca
Zhuangyuan (Bob) Shen	400354488	shenz57@mcmaster.ca

Table of Contents

Problem Statement and understanding 3

Data pre-processing 5

Approaches to implement 8

Problem Statement and understanding

Fake news consists of false information or deliberate made-up “facts” spread via means like traditional news networks, newspapers or nowadays more so on social media. Whether the fake news is artificially made or generated by an AI program, both have negative consequences. In the current internet age, fake news is all over the place and they could be quite deceiving and hard to differentiate from the real ones at times. Facebook had long been suffering from the consequences of fake news, not only was it accused of impacting the US presidential elections, it was also fined by the German government by 2 million euros in 2019. According to a Pew Research Center poll from December 2016, 23% of adults in the United States have shared fake news with friends and others, whether consciously or unwittingly. On a societal level, fake news could sabotage the stability of the society, causing tensions and distrust among people and groups. On an individual level, a piece of health-related fake news could have false information that actually does harm to an individual's health rather than helping it. A recent (and still ongoing) example is the large amount of fake news related to COVID-19, many people poisoned themselves by taking bleach since they were informed by the fake news that it could protect them from COVID-19.

One reason that fake news plagues is that it is very time-consuming to classify fake news from real news if the job were to be performed by humans, not to mention the associated accuracy and cost of doing so. On the other hand, even with the help of an AI program, censoring all the fake news is impossible because the speed of information generated grows exponentially, especially with the help of social media, people could post almost whatever (many of which are hearsay or deliberate false information) on it anytime and most of the posts went uncensored. Yet the speed of even using the best AI programs to scrutinize the information does not match. Additionally, the problems that deals with human language, known as Natural Language Processing (NLP) are one of the most challenging subsets of Machine Learning. The reason being while it is easy to represent an image in terms of a matrix, it is not as easy to encode texts as a number or a vector. Not to mention that NLP often needs to explore the way humans communicate and our consciousness.

The project team is tasked to leverage the data in the **Fake News dataset** with over 7k pieces of news belonging to one of the two classes - real or fake, and the main feature of each data point is **Text** column in the csv file, then make predictions about the authenticity of the news (Label column in the csv file). We will use a variety of machine learning and deep learning algorithms to build models and train classifiers to predict the results. By implementing these algorithms, compare the pros and cons of each algorithm, and evaluate the effect of each algorithm.

In conclusion, for this NLP binary classification problem, first preprocess the data set (see the second part of this report for details), and then use different machine learning and deep learning algorithms (see the third part of this report for details) to train the classifier. Make predictions and

evaluations, and finally analyze and compare different algorithms, to get a deeper understanding. For machine learning algorithms, we may use the sklearn library to implement, and use Tensorflow or Keras to implement deep learning models, and Pytorch may also be used if we need it in later development. In the data preprocessing stage, we will also import the NLTK library to achieve.

The codes can be found in below GitHub repo:
https://github.com/SLAM-CROC/Fake-news-detection

The project outcomes are:

- Pre-process the data to remove stop words. Stop words are the most occurring words in the language. It's necessary to filter that out first.

- Evaluate various algorithms which can affect best outcome

- Train a model to predict the likelihood of real news.

Project Timeline:

Oct 16th - Launch date

Oct 16th - Oct 17 - Interpret the problem, researching, selecting development tool and framework

Oct 18th - Oct 20 - Preprocessing the data & coding

Oct 21st - Oct 23 - Choice of approach to implement

Oct 24th - Report1&2: Problem statement and understanding, Data pre-processing and choice of approach to implement

Nov 7th - Report3: Explanation of the method(s) and competing approaches

Dec 5th - Class Presentation

Dec 12th - Final Report: Experimental evaluation and results % Class Presentation

Data pre-processing

For data pre-processing, we used regular expressions, Keras, and the NLTK language processing library (because stemming and lemmatization is not implemented in Keras Tokenizer), code structure can be seen by following screenshot:

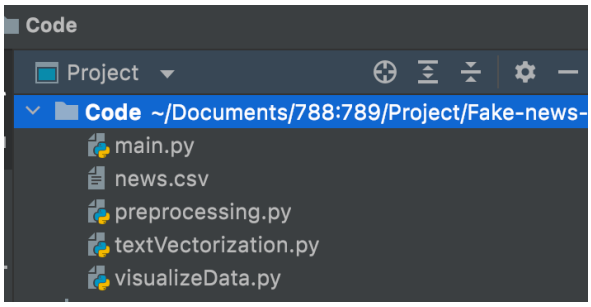

Fig1. Data Preprocessing code structure

We used a variety of different methods and steps. We encapsulated these methods in the **preprocessing.py** and manipulated the data by calling different methods. We can add or ignore some of these methods according to the different needs of the later training model stage, and we can also compare the impact of different data pre-processing methods on the prediction effect. And figure out which steps or methods are better.
We have implemented the following methods in **preprocessing.py**

1. Load dataset from csv file and delete the missing data points

   features_data, labels_data = preprocessing.load_csv('news.csv')

2. Remove URL in the text, there some URL exist in the text, under normal circumstances, it has little effect on the meaning of language, we can choose to remove it from our data

   preprocessing.remove_url(features_data)

3. Remove newline signals from text

   preprocessing.remove_newline(features_data)

4. Remove numbers from text

   preprocessing.remove_number(features_data)

5. Remove punctuations from text

preprocessing.remove_punctuation(features_data)

6. Convert all letters into lowercase

   preprocessing.convert_into_lowercase(features_data)

7. Tokenization text (Implemented by Python NLTK package)

   preprocessing.tokenization(features_data)

8. Remove stop words from text (Implemented by Python NLTK package)

   preprocessing.remove_stopwords(features_data)

9. Normalization: including stemming and Lemmatization ()

   preprocessing.normalization(features_data)

10. Remove words whose length are equal or less than 2, because most words with less than two letters are meaningless

    preprocessing.remove_short_words(features_data)

So far by calling the function in **preprocessing.py**, we have got the cleaned text.

Then we can use python's drawing tools (**Matplotlib**) to visualize the data, so as to take a look at our data and increase our understanding of it. We implemented these in the **visualizeData.py**. Finally, we can use different text vectorization methods for the final processing of the text. We will implement different methods in the **textVectorization.py**, and currently we use the keras' built-in Tokenizer of keras.

We drew the following word frequency distribution histogram:

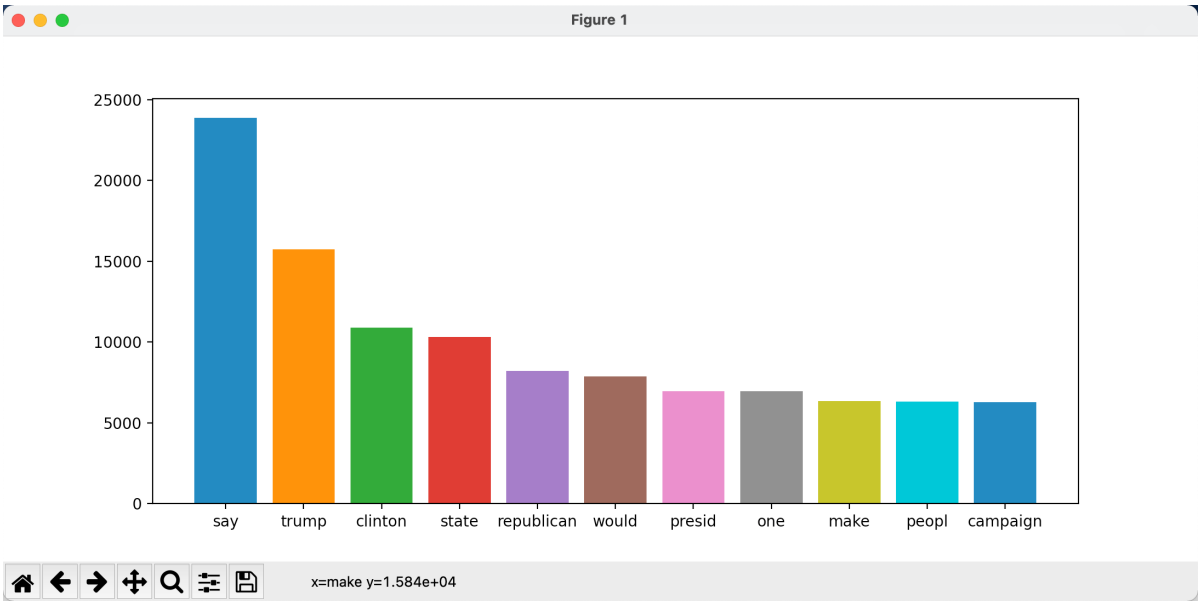

Fig2. Word frequency distribution histogram for Real news

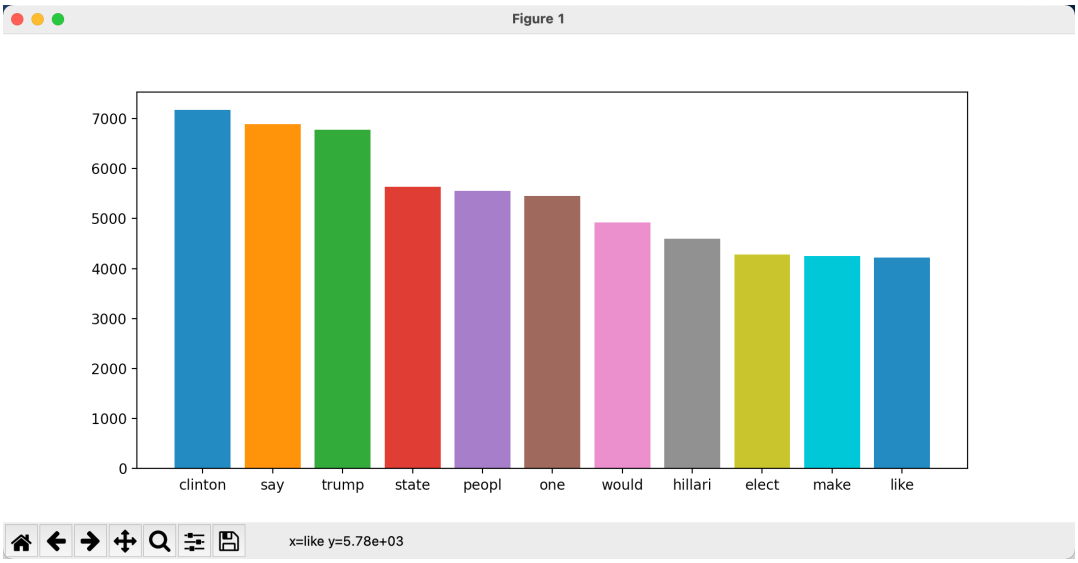

Fig2. Word frequency distribution histogram for Real news

Text vectorization/Word Embedding

Currently we use the keras' built-in Tokenizer of keras to implement the Text vectorization/Word Embedding. Take an example, after preprocessing original text:

"Jeb Bush Is Suddenly Attacking Trump. Here's Why That Matters

Jeb Bush isn't pulling punches anymore when it comes to Donald Trump.

The former Florida governor has delicately danced around the billionaire businessman in the 2016 presidential primary so far. But the gloves came off this week when Bush called out Trump as a closet Democrat. He was trying to stunt Trump's rise while attempting to recover his own political mojo."

Will become:

[5, 2, 69, 4, 1, 70, 5, 2, 24, 16, 71, 25, 26, 1, 9, 72, 27, 28, 73, 74, 75, 76, 77, 78, 29, 30, 25, 17, 2, 18, 1, 79, 10, 11, 80, 1, 31, 81, 82, 32, 83, 5, 84, 11, 85, 86, 87, 6, 88, 89, 33, 3, 90, 5, 34, 91, 92, 93, 94, 95, 96, 97, 98, 35, 36, 99, 37, 2, 38, 19, 100, 101, 102, 4, 1, 28, 103, 39, 104, 40, 41, 105, 106, 107, 4, 108, 109, 110, 111, 12, 112, 42, 20, 2, 113, 114, 115, 18, 1, 116, 117, 3, 1, 118, 119, 1, 18, 2, 120, 1]

Approaches to implement

There are several classification algorithms for text and document classification, and we are going to discuss some machine learning and deep learning algorithms for this task. In the future development, we may make adjustments to this part, mainly to add or subdivide each model。

1. Machine Learning Algorithm

1.1. Logistic Regression

Logistic Regression is a linear classifier that predicts class probabilities.

1.2. Naive Bayes Classifier

Naive Bayes Classifier is a widely used generative model for document categorization which is theoretically based on Bayes theorem. Bag-of-words lays the foundation for the basic form of Naive Bayes Classifier. The Naive Bayes algorithm can be described:

$$P(c|d) = (P(d|c)P(c))/P(d)$$

where d denotes the document and c indicates classes and P gives the posterior probability.

1.3. K-Nearest Neighbour

The k-Nearest Neighbours algorithm is a non-parametric technique used for classification. The basic idea of the kNN classifier is finding the k nearest neighbours of the candidate document from among the training documents and scoring the categories of the k nearest neighbours. The candidate document is thus assigned to the class with the highest score.

1.4. Support Vector Machine

Support Vector Machine (SVM) is the supervised learning algorithm for classification and regression. The main objective of SVM is to find a maximal separating hyperplane between vectors that belong to a category and vectors that do not belong to it. The maximum distance between data points helps to classify future data points with more confidence.

1.5. Decision Tree

A decision tree is a tree-like structure that classifies the data by tracing the path from root to leaf. The attribute with the largest information gain is chosen as the parent node and the subsequent attributes are assigned to the child node.

2. Deep Learning Algorithm

2.1. Recurrent Neural Networks (RNN)

Recurrent Neural Network (RNN) is another neural network architecture widely used for text classification problems. Because of the high-dimensional hidden state they can process past information thus making the architecture very useful for text and sequential data classification.

2.2. Long Short-Term Memory (LSTM)

LSTM addresses the problems of vanishing gradient by preserving long term dependency more effectively. LSTM works with multiple switch gates which help regulate the amount of information entering each node.

2.3. Convolutional Neural Networks (CNN)

Another deep learning architecture that has been in practice for text classification is Convolutional Neural Network (CNN). Despite its popularity for image processing, CNN has been able to outperform different text classification models in recent times.