Fake News Detection using a BERT-based Deep Learning

Introduction

Fake News is pervasive nowadays and is too easy to spread with social media and it is difficult for us to identify. Since the Covid-19 outbreak in Taiwan recently, a lot of fake news has popped up on social networks, such as LINE, Facebook, PTT (one of the largest online forums in Taiwan), etc. Hence, we aim to utilize multiple artificial intelligence algorithms to detect fake news to help people recognize it.

Related Work

In this section, we discuss some previous work that is related to fake news detection. Fake news can be defined as fabricated information that mimics news media content in form but not in organizational process or intent [1].
In recent years, a lot of automated fake news detection methods have been proposed. For example, Shu, Kai, et al. [2] provided numerous methods to solve the problem of fake news classification, such as user-based, knowledge-based, social network-based, style-based methods, etc. Julio, et al. [3] presented a new set of features and measured the prediction performance of current approaches and features for automatic detection of fake news. Daniel, et al. [4] focused on the analysis of information credibility on Twitter. Heejung, et al. [5] applied the Bidirectional Encoder Representations from Transformers model (BERT) model to detect fake news by analyzing the relationship between the headline and the body text of news.
Mohammad Hadi, et al. [6] applied different levels of n-grams for feature extraction based on the ISOT fake news dataset. Saqib, et al. [7] proposed an ensemble classification model for the detection of fake news that has achieved a better accuracy compared to the state-of-the-art also on the ISOT fake news dataset. Sebastian, et al. [8] used a neural network-based approach to perform text analysis and fake news detection on ISOT fake news dataset as well.

Methodology

The process of fake news detection can be divided into four stages - data preprocessing, word embedding, models, and model fine-tuning.

The Dataset

Link: https://www.uvic.ca/engineering/ece/isot/datasets/fake-news/index.php

The dataset we use is the ISOT Fake News dataset introduced by ISOT Research Lab at University of Victoria in Canada [9]. This dataset is a compilation of several thousand fake news and truthful articles, obtained from different legitimate news sites and sites flagged as unreliable by Politifact.com. To get insight into this dataset, we visualized it with word clouds for real and fake news respectively. Figure 1(a). shows the word cloud of the real news in the dataset, and Figure 1(b). shows the one of the fake news in the dataset.

Figure 1(a).

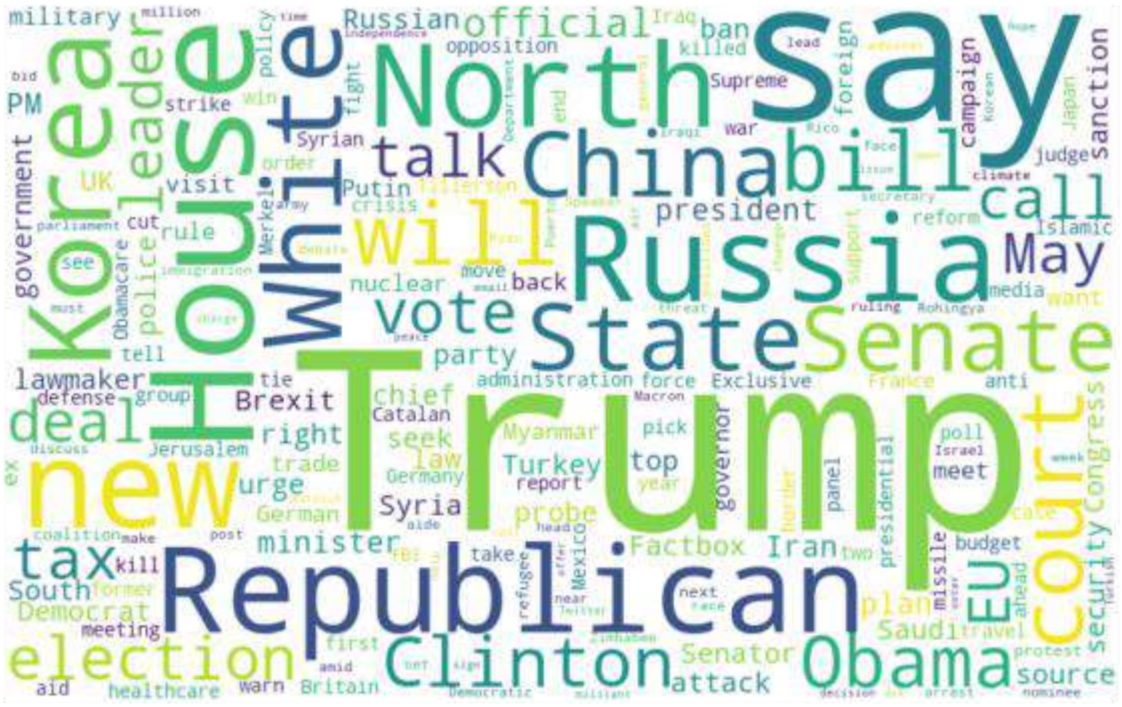
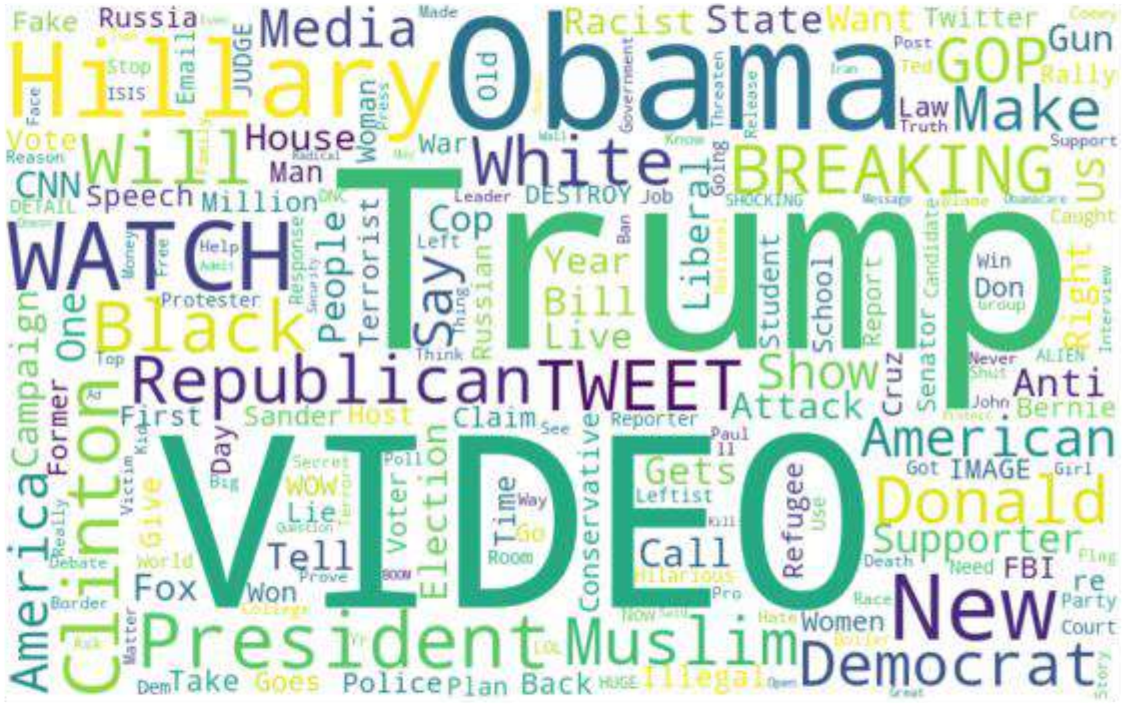


Figure 1(b).



We can see that, in the real news word cloud, 'Trump', 'say', 'Russia', 'House', 'North', and Korea' appeared frequently; while in fake news one, 'VIDEO', 'Trump', 'Obama', 'WATCH', and 'Hillary' appeared the most frequently. 'Say' appears frequently in real news but does not in fake news. 'VIDEO', and 'WATCH' appear frequently in fake news but do not in real news. From these two word clouds, we can get some important information to differentiate the two classes of data. The original form of the dataset is two CSV files containing fake and real news respectively. We combined the dataset and split it into training, validation, and test sets with shuffling at the ratio of 64%:16%:20%. The original combined dataset contains 44,898 pieces of data, and Table 1. shows the distribution of data in the training, validation, and test sets.

Table 1. Distribution of Data

Training	Validation	Test
64%	16%	20%
28734	7184	8980

Data Preprocessing

The main goal of this part is to use NLP techniques to preprocess the input data and prepare for the next step to extract the proper features.
The data we use contains news titles and texts. Each of the titles is about 12.45 words long, while each of the texts is about 405.28 words long. In our project, we only use the titles for the fake news detection because the texts are too large for us to train efficiently. Also, the text contains too many details and information for a piece of news, which may distract the models during training.
We built a preprocessing pipeline for each statement to eliminate the noise in the fake news dataset. The preprocessing pipeline includes the following 3 sub-parts:

1. Replaced characters that are not between a to z or A to Z with whitespace.
2. Converted all characters into lower-case ones.
3. Removed the inflectional morphemes like "ed", "est", "s", and "ing" from their token stem. Ex: confirmed → "confirm" + " -ed"

We also cropped the titles into sentences with a maximum length of 42 in order to train the model on a dataset with sentences of reasonable lengths, and also eliminate titles with an extreme length that may let the model fit on unbalanced data.

Word Embedding

This part is important because we need to convert the dataset into a form that models can handle. We use different types of word embedding for different models we built. For LSTM, Bidirectional LSTM, and CNN, we first create a Tokenizer to tokenize The words and create sequences of tokenized words. Next, we zero-padded each sequence to make the length of it 42. Then, we utilized the Embedding layer that initialized with random weights to let it learn an embedding for all of the words in the training dataset. The Embedding layer [10] will convert the sequence into a distributed representation, which is a sequence of dense, real-valued vectors. For BERT, we utilized BERT tokenizer to tokenize the news titles in the dataset first. BERT uses Google NMT's WordPiece Tokenization [11] to separate the words into smaller pieces to deal with words that are not contained in the dictionary. For example, embedding is divided into ['em', '##bed', '##ding'].
Also, there are two other special tokens that BERT needed, which are [CLS] and [SEP]. [CLS] is put at the beginning of a sentence representing the front of an input series. [SEP] is put at the middle of two sentences when they are combined to a single input series, or put at the end of a sentence if the input series is only one sentence.

Then, for the input of BERT, we need to convert the original statements into three kinds of tensors, which are token tensors, segment tensors, mask tensors. Token tensors represent the indices of tokens, which are obtained by the tokenizer. Segment tensors represent the identification of different sentences. Mask tensors represent the concentration of tokens including information after zero-padding the data into the same length.

Models

- **LSTM**
  Long-Short Term Memory (LSTM) is an advanced version of Recurrent Neural Network (RNN), which makes it easier to remember past data in memory. LSTM is a well-suited model for sequential data, such as data for NLP problems. Thus, we utilized LSTM to perform fake news detection.

- **Bidirectional LSTM**
  Bidirectional LSTM (BiLSTM) [12] consists of two LSTMs: one taking the input from a forward direction, and the other in a backward direction. BiLSTM effectively increases the amount of information available to the network, improving the context available to the algorithm.

- **CNN-BiLSTM**
  In this section, we used Convolutional Neural Network (CNN) as the upper layer of the bidirectional LSTM. That is, the output of the CNN is the input of the BiLSTM. This architecture extracts the maximum number of features and information of the input text with convolutional layers, and also utilizes the bidirectional benefit of BiLSTM to ensure that the network can output based on its entire input text.

- **BERT**
  In the Natural Language Processing field, Transformers become more and more dominant. BERT [13], the acronym for Bidirectional Encoder Representations from Transformers, is a transformer-based machine learning technique that changed the NLP world in recent years due to its state-of-the-art performance. Its two main features are that it is a deep transformer model so that it can process lengthy sentences effectively using the 'attention' mechanism, and it is bidirectional so that it will output based on the entire input sentence.
  We used BERT to handle the dataset and construct a deep learning model by fine-tuning the bert-based-uncased pre-trained model for fake news detection. Training a model for natural language processing is costly and time-consuming

because of the large number of parameters. Fortunately, we have pre-trained models of BERT that enable us to conduct transfer learning efficiently. We choose the pre-trained model of bert-base-uncased from a lot of models with different kinds of parameters. The chosen one consists of a base amount of parameters and does not consider cases of letters (upper-case and lower-case).

Model Fine-Tuning

For different downstream tasks, we need to conduct different fine-tuning approaches. Thanks to HuggingFace, we have the models for different downstream tasks. In our project of fake news detection, which is the classification of statements, we used bertForSequenceClassification to fine-tune our pre-trained BERT model. The modules of the model contain a BERT module handling various embeddings, a BERT transformer encoder, a BertPooler, a dropout layer, and a linear classifier that returns logits of the 2 classes.

Experiments

Imbalanced Data

Our dataset is balanced between real news and fake news. However, in the real world, real news and fake news are not as balanced as what the dataset shows. We assumed that the amount of real news is way larger than the amount of fake news to reflect the real-life situation. This, we did two experiments for each model with balanced and imbalanced datasets respectively to compare the performance between them. To build the imbalanced dataset, we shrunk the original fake dataset into one-tenth of the original size to imitate the real-world situation. Table 2. shows the true/fake distribution of data in the original training set, imbalanced training set, original validation set, imbalanced validation set, original test set, and imbalanced test set.

Data	**Original Training Set**	**Imbalanced Training Set**	**Original Validation Set**	**Imbalanced Validation Set**	**Original Test Set**	**Imbalanced Test Set**
True	13765	13765	3409	3409	4243	4243

Data	Original Training Set	Imbalanced Training Set	Original Validation Set	Imbalanced Validation Set	Original Test Set	Imbalanced Test Set
Fake	14969	1497	3775	378	4737	474

Results

To evaluate the performance of the models we constructed for the fake news detection problem, we used the most commonly used metrics:

- True Positive (TP): when predicted fake news is actually fake news
- True Negative (TN): when predicted true news is actually true news
- False Negative (FN): when predicted true news is actually fake news
- False Positive (FP): when predicted fake news is actually true news

We can define the following metrics based on the value of the above 4 situations.

$$
\begin{aligned}
Precision &= \frac{|TP|}{|TP| + |FP|} \\
Recall &= \frac{|TP|}{|TP| + |FN|} \\
F1 &= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \\
Accuracy &= \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}
\end{aligned}
$$

These 4 metrics are the most widely used in the world of machine learning, especially for classification problems. It allows us to evaluate the performance of a classifier from different perspectives. Normally, ‘Accuracy’ is the most representative metric for the evaluation because it reflects the situation of classification completely.

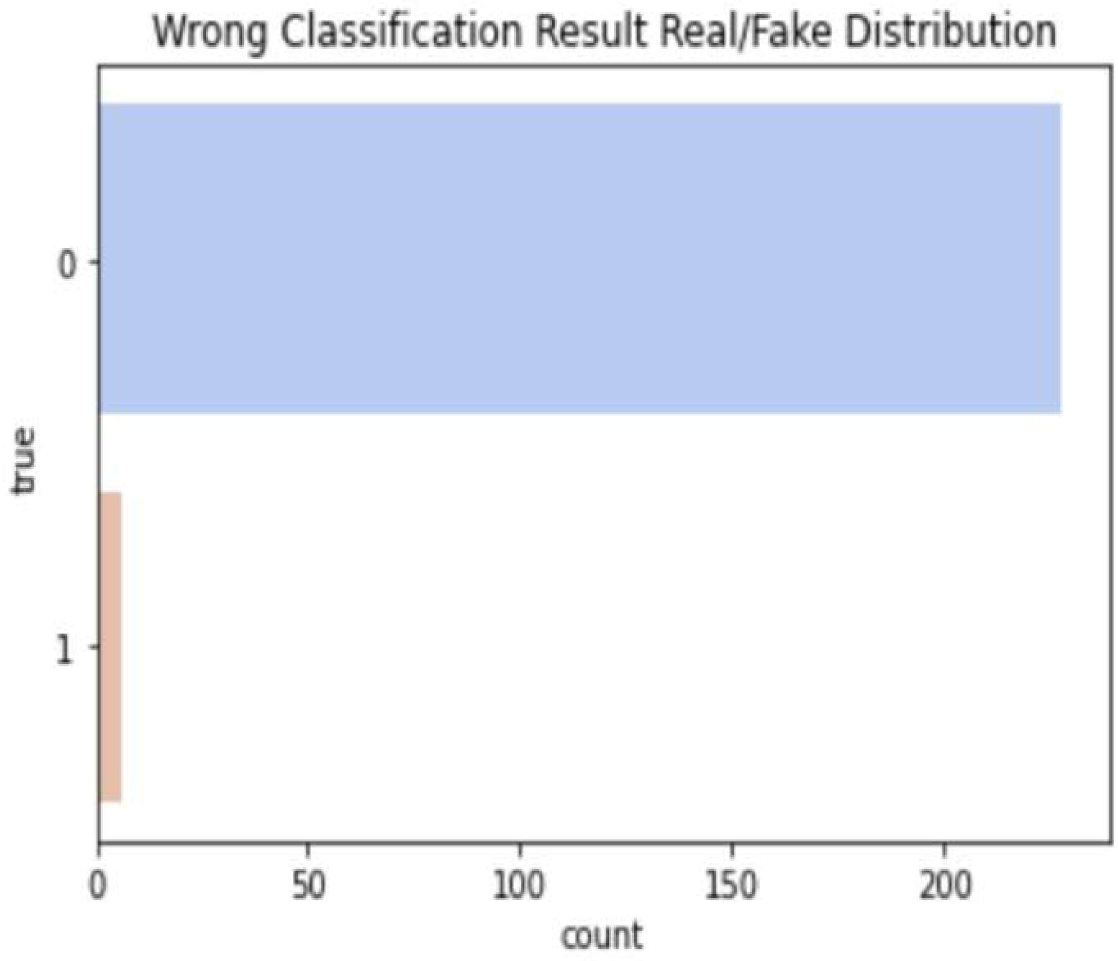
On Balanced Dataset

Model	Accuracy	Precision	Recall	F1-Score
LSTM	0.9697	0.97	0.97	0.97
BiLSTM	0.9710	0.97	0.97	0.97
C-BiL	0.9722	0.97	0.97	0.97
BERT	0.9874	0.99	0.99	0.99

On Imbalanced Dataset

Model	Accuracy	Precision	Recall	F1-Score
LSTM	0.9780	0.98	0.98	0.98
BiLSTM	0.9799	0.98	0.98	0.98
C-BiL	0.9791	0.98	0.98	0.98
BERT	0.9903	0.99	0.99	0.99

Figure 2. Class Distribution of wrongly classified data


Wrong Classification Result Real/Fake Distribution
true
count