Round 2 Report and Reflection

Akmarzhan Abylay

CP191 Fall 2020

Abstract

This project uses a neural network that identifies the critical features of text material (i.e., the distribution of words in the text) to distinguish fake news from real news. I will pre-process the text data I have (e.g., clean from stopwords, remove punctuation) and implement a feedforward[1] neural network that will maximize the accuracy of predictions through penalizing the binary cross-entropy score explained further in the paper. I will present a working code and visualizations of the result and include a brief description of the used methods.

Word count: 88

Introduction

Fake news is the content that includes untruthful information with the goal of misleading readers, gaining viral attention, or damaging one's reputation ("Fake news," n.d.). It is turning into a massive problem in today's society as more and more information is being falsified, and it is becoming easier to share and spread fake news due to technological improvements. It has even been observed that tweets with fake information spread 6 times faster than the real ones (Langin, 2018). The study also suggests that we might extrapolate that the same trend is true for other platforms, such as Facebook, Whatsapp, Instagram, Telegram, etc.

The impact of fake news ranges from making people think that Morgan Freeman is dead when he isn't and convincing readers that Hillary Clinton gave birth to an alien baby, all the way to brainwashing the older population into believing that COVID-19 is a hoax. These are just examples of how ridiculous fake news can be, but the main problem is that people still believe in them, even if they seem untrustworthy at first sight. The difficulty of spotting fake news is rising

[1] A feedforward neural network is a type of neural network that passes the input forward to other nodes and has no cyclical node connections.

over time, and the misinformation can get to an even higher level, which is affecting many human lives. For example, there has been a hoax in Kazakhstan that kids can't get COVID-19, vodka kills the virus, and people can die from a vaccine against COVID-19 (Kim, 2020). In uncertain situations, it can get tough to spot if the information is false, which is why I want to help stop the misguidance. We can use computational tools to do this. Specifically, data preprocessing steps and neural networks (NN) can help us. There has already been some research in the field and many scientists have obtained high (i.e., above 95%) accuracy rates (explained further in the text). They mostly used recurrent NNs, which is why I want to test out how possible this result is using feedforward NNs.

The task's main difficulty is that it is extremely hard to detect whether the information is fake, and the choice can be highly subjective. Due to the scope of this project, I will assume that the dataset I found objectively classified the news into fake and real, and we will use the accuracy (i.e., correctly found labels - "real" or "fake") of the neural network as the metric for success.

Word count: 346

Work Products

I have created an algorithm for fake news classification using the dataset obtained from Kaggle (“Fake news,” 2017). I had an assumption that my dataset had correctly labeled entries (more than 20000), so I used them for training and testing my neural network model. I first pre-processed the data by removing all punctuation, capital letters, and common English words (e.g., is, it, he, she, etc.). After additional processing (i.e., modifying the input to fit the model), I

created a neural network inspired by a paper, "Fake News Identification," from Stanford University (Mone, Devyani & Singhania, n.d.).

I initially split the dataset into training and testing sets, and I trained the model on the training data, so the test set here refers to data that my model hasn't seen. This is an appropriate measure as it ensures that the accuracy rate we obtain is unbiased, and we are not checking the results on the data the model has already seen. I also used a loss function called binary cross-entropy since we are doing binary classification and our target outputs are either 0s or 1s (i.e., real data or fake data). The loss function minimizes the difference between the predicted and actual probabilities (i.e., for true labels, it is either 0 or 1, though), so it is suitable for the classification we are trying to perform. For context, 0 is the perfect score because it means that the algorithm correctly identified the label. The loss function helps our neural network know how good it is performing and improving during the iterations.

The easiest way to think about the NN output is that the input is going through several layers of the NN and in the end, we perform one more layer of classification, which is very similar to how logistic regression (i.e., that uses a logistic function to predict a binary dependent variable) works. In our case, we output probabilities of the observation being positive (i.e., fake) though and based on the chosen threshold (i.e., 50%, where if the probability is higher than 0.5, it is classified as fake), we classify the observations. Thus, we could obtain the same results by using the neural network for extracting the most important features (i.e., pre-processing the input) and feed them further into a logistic regression classifier that outputs binary dependent variables (i.e., 0 is real news, 1 is fake news)[2].

[2] **#regression:** accurately interpreted a regression model with a well-justified explanation of the relationship between the dependent and independent variables (i.e., the last layer of the neural network being used as almost a logistic regression tool which inputs the NN-processed features and outputs a binary variable defining whether the information is fake or real).

In the end, I obtained a 96.11% training set accuracy and a 92.15% testing set accuracy. As mentioned, the model hasn't seen the test set and was still able to correctly label more than 90% of the texts (see Appendix for code and results). Below I plotted the true labels vs. the predicted labels from the model (i.e., confusion matrix).

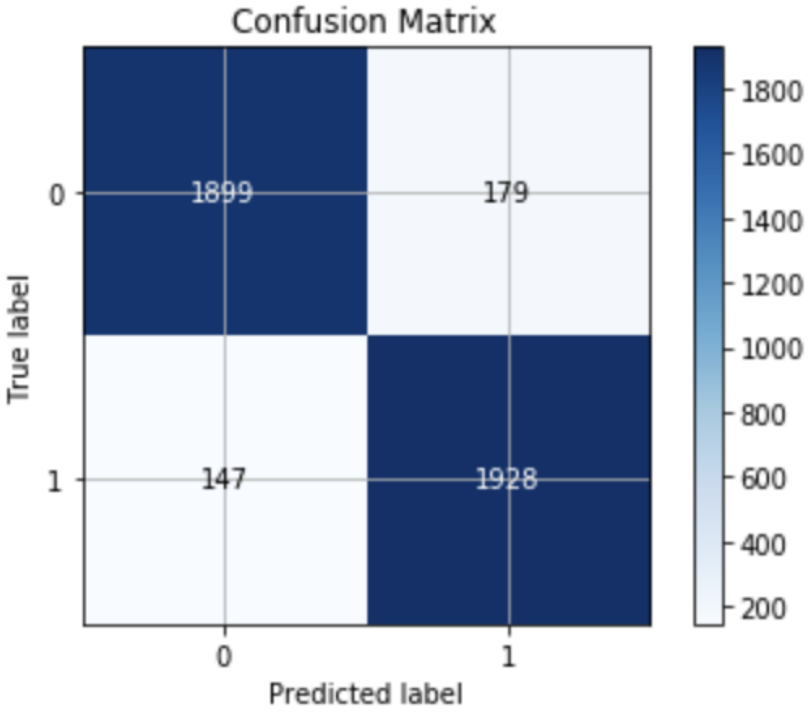

Figure 1. Confusion matrix for the neural network classifier.

The data is balanced (i.e., there is an equal number of fake and real news), and the confusion matrix showed us a balanced prediction. This means that there is around the same number of True Positives (i.e., fake news classified as fake by the model) and True Negatives (i.e., real news classified as real by the model) and the same for the False Positives (i.e., real news classified as fake by the model) and False Negatives (i.e., fake news classified as real by the model)[3].

[3] **#metrics:** I developed appropriate measures to evaluate my work (i.e., accuracy rate, binary cross-entropy score, confusion matrix); comprehensively classified the quality and completeness of work products and identified important areas for improvement (i.e., last paragraph of Quality of Deliverables).

After pre-processing, there were around 20260 data points in the training set. The data was trained on 80% of it and showed excellent results - approximately 92% test set accuracy. We can still improve the model by training on more data or adding more pre-processing.

Word count: 606

Analysis and Reflection

Quality of Deliverables

Similar to what I did for Round 1 Report and Reflection, my main deliverable was working code. It is suitable for CS-driven projects because it is an MVP, which provides enough information for the users to give feedback in the early stages. The scope was right as I simplified the problem and used pre-existing methods to obtain an MVP instead of investing much more time in creating the final product from scratch[4].

I used the test set accuracy as the metric for how well my model does, as it represents the fraction of correctly classified data points over the length of the test set. As mentioned, I also use a binary cross-entropy loss function, which is suitable for probability-based outputs. Both of them are appropriate metrics for the classification as they show us exactly how good our model is and where we need improvements.

As mentioned, the code is still in draft-mode and requires more training and better methods. The next step is to try out different Natural Language Processing (NLP) techniques and various layer configurations for the neural network to see if the accuracy could be improved. It is already around 92%, but this is under the assumption that data is correctly labeled in the first place. We can test the model on a completely different set of texts to identify how well it

[4] **#qualitydeliverables**: submitted an MVP, which is appropriate for this round of exploration; justified why it is appropriate (i.e., the first paragraph of "Quality of Deliverables").

performs and add more features to the pre-processing, such as lemmatization[5].

Approach and Process

I used techniques covered in my machine learning concentration course CS156 and a methodology from the papers mentioned above. Similar to Round 1, a literature review was essential as I wouldn't have been able to build a classifier without learning the basics of pre-processing the text and using neural networks.

In my Round 2 Project Exploration, I mentioned that I wanted to do a literature review/research on NLP and neural networks and possibly create working code. My progress on this assignment differed because I actually made an MVP (i.e., code), and I only briefly touched upon the results from the literature review. Also, I was planning on using recurrent neural networks (i.e., LSTM - Long Short-Term Memory), but I ended up implementing a classical feedforward NN. Surprisingly, writing the code for this task wasn't as difficult as I thought it would be in the beginning, which is why it took me less time to complete. Based on my Round 1 Project Exploration, I understood what I am capable of doing in a limited amount of time, which is why I was able to follow through with what I wrote in the Focus for Round 2.

I made a working code using libraries, such as "TensorFlow" and "sklearn," which demonstrated that I could complete a similar project. I already made an MVP, so it will be experimenting and improving the original draft from now on if I decide to continue this as a Capstone. If I were to start again, I would think about the project's limitations in more detail because it turned out to be extremely hard to find a relevant dataset for this problem due to uncertainty in the truthfulness of the information.

[5] We also could take more factors into account and classify the fake news into more types, such as false content, misleading content, satire, manipulated content, and many more ("Fake news," n.d.).

Potential as a Capstone

Although the rigor and scope were appropriate for this project, after working on both project exploration rounds, I concluded that I enjoy working on computer vision-related problems more. This project closely relates to my major though (i.e., CS), and I could apply many of the relevant LOs[6]. However, as mentioned above, I realized that I am more interested in computer vision applications, so I wouldn't pursue this project as a Capstone.

Next Steps

Based on my experience, I think I will not pursue Round 1 or Round 2 projects as my Capstone. The issue with Round 1 was that the problem I was trying to solve (i.e., style transfer) wasn't critical, and I wasn't that motivated to continue working on it. For Round 2, although the topic was quite enjoyable, it didn't involve computer vision algorithms. Since I couldn't visualize the results (i.e., I only had a confusion matrix), it was not as fascinating compared to when I saw the processed photos in the styles of my favorite artists (i.e., Round 1)[7].

I will look for another Capstone idea during the winter break and think in advance about all the things I didn't consider in the previous rounds of project exploration (e.g., managing time, motivation, limitations in terms of the available data). I will also try to incorporate my interests into the Capstone (e.g., computer vision; hardware incorporation, such as Raspberry Pi)[8].

Word count: 710

---

[6] LOs, such as #cs156-neuralnetworks (i.e., implementing NNs), #cs156-modelmetrics (i.e., for choosing appropriate metrics), #cs164-optimizationstrategy (i.e., optimizing the algorithm so that it performs better), and many more.

[7] **#discovery:** defined and explored a problem in a thorough manner (i.e., discovered how to perform fake news detection using the relevant tools) using high-quality sources (i.e., mentioned Stanford University research paper) to identify a worthwhile focus for a project (i.e., recognized that I am not as interested in an NLP problem, and prefer working on visual perception field more).

[8] **#navigation:** applied an appropriate mindset to enable successful completion of a project (e.g., I used a growth mindset to learn new NLP skills and develop a neural network required to complete the project; I also found out that I am more interested in computer vision-related problems).

Appendix

Full code can also be found here. The dataset can be found here.

```
#importing the libraries
import numpy as np
import re
import string
import pandas as pd
from gensim.models import Doc2Vec
from gensim.models.doc2vec import LabeledSentence
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
```

```
def clean(doc):
    """
    Cleaning the text: removing punctuation, stopwords and
    making them all lower letter.
    """

    doc = re.sub(r"[^A-Za-z0-9^]", " ", doc)
    doc = [word for word in doc.lower().split() if not word in stopwords.words("english")]
    doc = " ".join(doc)
    return doc

def labels(data):
    tags = []
    for index, row in enumerate(data):
        tags.append(LabeledSentence(row.split(), ['doc_%d'% index]))
    return tags

```

```
def pre_process(path, size=300):
    """
    Function for pre-processing using Doc2Vec.
    """

    data = pd.read_csv(path)
    data = data.dropna(subset=['text']).reset_index().drop(['index', 'id'], axis=1)

    data['text'] = data['text'].apply(lambda row: clean(row))

    X = labels(data['text'])
    y = data['label'].values

    model_t = Doc2Vec(min_count=1, vector_size = size, window=5, sample=1e-4, negative=5,
                      workers=10, epochs=10, seed=0)
    model_t.build_vocab(X)
    model_t.train(X, total_examples=model_t.corpus_count, epochs=model_t.iter)

    X_train, X_test, y_train, y_test = train_test_split(model_t.docvecs, y, test_size=0.2,
                                                        random_state=0, stratify=y)
    X_train, X_test, y_train, y_test = np.array(X_train), np.array(X_test),
                                       np.array(y_train), np.array(y_test)

    return X_train, X_test, y_train, y_test
```

```
import matplotlib.pyplot as plt
import keras
from keras import backend as K
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, Embedding, Input, RepeatVector
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
import scikitplot.plotters as skplt
import os

X_train, X_test, y_train, y_test = pre_process('train.csv')
```

```
def model():
    '''Initializing the neural network.'''
    model = Sequential()
    model.add(Dense(256, input_dim=300, activation='relu', kernel_initializer='normal'))
    model.add(Dropout(0.3))
    model.add(Dense(256, activation='relu', kernel_initializer='normal'))
    model.add(Dropout(0.5))
    model.add(Dense(80, activation='relu', kernel_initializer='normal'))
    model.add(Dense(1, activation="sigmoid", kernel_initializer='normal'))

    #compiling a gradient descent algorithm
    model.compile(loss='binary_crossentropy', optimizer= Adam(lr=0.01), metrics='accuracy')
    return model

#setting the model
model = model()
```

```
estimator = model.fit(X_train, y_train, epochs=20, batch_size=64, verbose=0)
print("Model Trained!")
```

```
train = model.evaluate(X_train, y_train)
test = model.evaluate(X_test, y_test)

print("Train Set Accuracy: {}\nTest Set Accuracy: {} ".format(str(round(train[1]*100, 2)),
                                                              str(round(test[1]*100, 3))))
```

```
519/519 [==============================] - 0s 737us/step - loss: 0.0995 - accuracy: 0.9611
130/130 [==============================] - 0s 900us/step - loss: 0.2280 - accuracy: 0.9215
Train Set Accuracy: 96.11
Test Set Accuracy: 92.15
```

```
y_pred = [0 if x<0.5 else 1 for x in model.predict(X_test)]
skplt.plot_confusion_matrix(y_test, y_pred)
```

References

Kim. S. (2020). Vodka kills COVID-19 in half a minute. Retrieved from https://factcheck.kz/v-mire/vodka-ubivaet-koronavirus-za-polminuty-novyj-fejk-gulyaet-v-socsetyax/

Langin, K. (2018). Fake news spreads faster than true news on Twitter—thanks to people, not bots. Retrieved from http://www.sciencemag.org

Mone, S., Choudhary, D. & Singhania, A. (n.d.). FAKE NEWS IDENTIFICATION: MACHINE LEARNING. Retrieved from http://cs229.stanford.edu/proj2017/final-reports/5244348.pdf

"Fake news." (n.d.) Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Fake_news

"Fake news". 2017. Kaggle. Retrieved from https://www.kaggle.com/c/fake-news/data