Fake News Detection Using Deep Recurrent Neural Networks

Pushpendra Singh Chauhan

Department of Computer Science

Illinois Institute of Technology

April 11, 2021

Abstract

This is a presentation on the final project. It will contain the description of the problem I am trying to solve, the algorithms I employed to solve the problem, implementation details like program design issues and the results obtained. The results obtained will be analyzed for correctness. The performance of the algorithm will be evaluated and discussed.

1. Problem Statement:

- Write a program in Python using TensorFlow and Keras libraries to perform fake news detection (i.e. predicting whether a news article is authentic or not) using bidirectional RNN/BiLSTM technique of deep learning.

2. Proposed Solution:

- **2.1 Dataset:** For this task I have used ISOT Fake News dataset. In this dataset, 44898 news articles are contained in total, 21417 of them are real and 23481 are fake. For real news articles label class is 1 and for fake news articles label class is 0. The dataset features include title, text, subject and date. To get enough information, we combined the ‘text’ and ‘title’ column for training and testing our model. All text data are cleaned and tokenized before feeding them into our BiLSTM model.

- **2.2 Bidirectional Recurrent Neural Networks / BiLSTM:** Bidirectional recurrent neural network performs better on some tasks than normal RNNs and usually is used in natural language processing. RNN relies on sequence or time series to capture pattern. Bidirectional RNNs can capture some patterns that are ignored by normal RNNs. A Bidirectional LSTM or BiLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, the other in a backward direction and combines their representations at last. BiLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence).

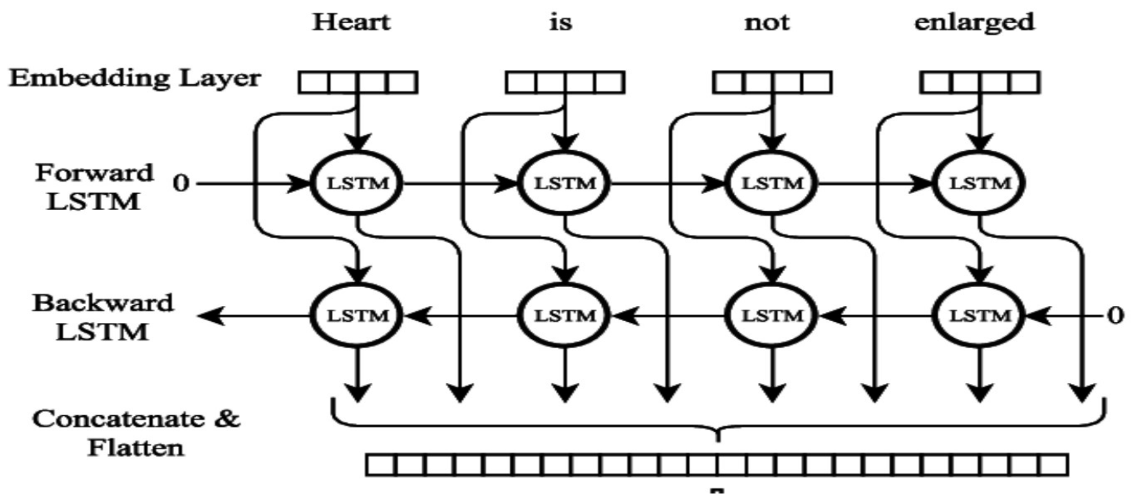
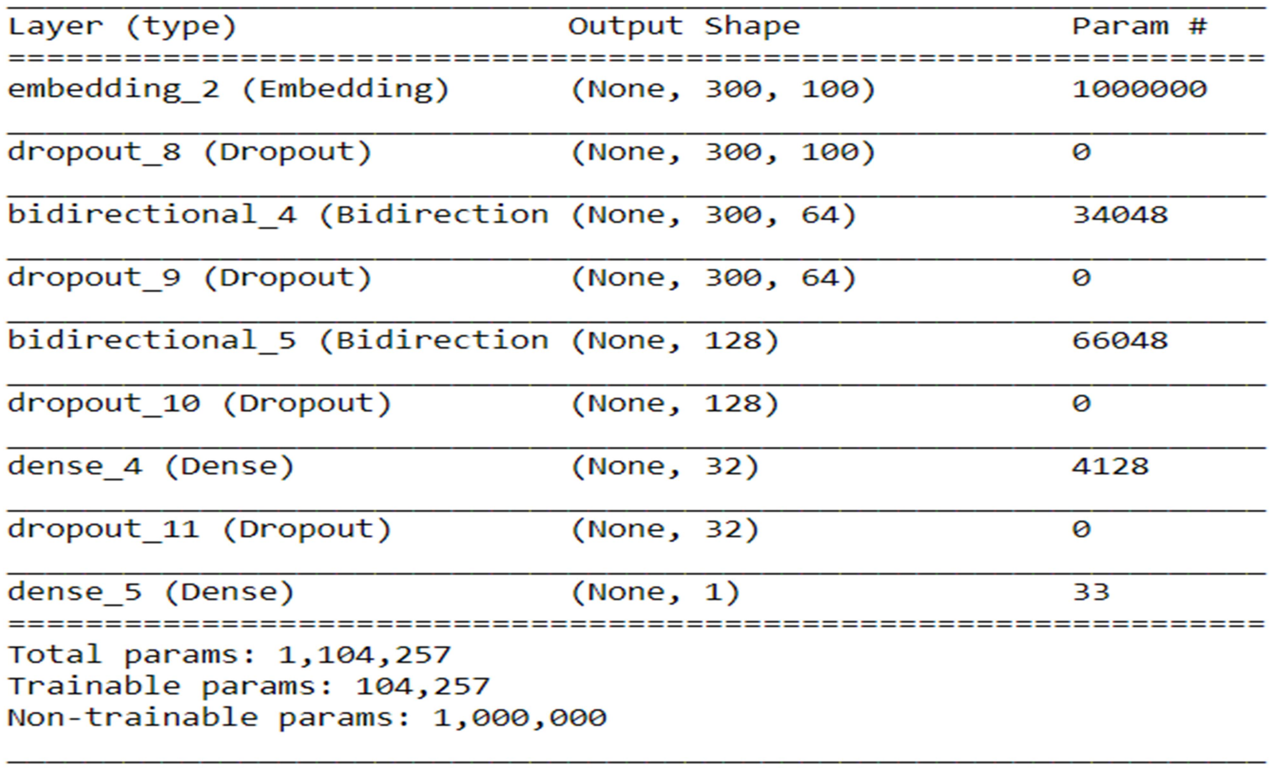


Figure: Bidirectional LSTM / BiLSTM

- **2.3 Text Preprocessing:** Text preprocessing usually means removal of stopwords, removal of punctuation marks, performing stemming, tokenization and weighting words in the document. To transform tokenized text into vectors, word embedding is often used. For word sequences, pretrained embedding neural networks like word2vec and glove are commonly used. In our project, we used glove to get the pretrained weighted word vectors. Considering the computation time, we limit the vocabulary to only consider the top 10000 terms across the entire corpus. We also padded every instance into 300 words long by using Keras.

- **2.4 Proposed Bidirectional LSTM/BiLSTM Neural Network architecture:** In the embedding layer, input_dim is the length of selected vocabulary, namely 10000. The input length is 300 just like the length of our padded instances. Output dimensionality is 100. To avoid overfitting, we also used dropout layer and 'L2' regularizer to reduce the parameters. Other detailed description of the BiLSTM neural network architecture is given in the figure below:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 300, 100)          1000000
_________________________________________________________________
dropout_8 (Dropout)          (None, 300, 100)          0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 300, 64)           34048
_________________________________________________________________
dropout_9 (Dropout)          (None, 300, 64)           0
_________________________________________________________________
bidirectional_5 (Bidirection (None, 128)               66048
_________________________________________________________________
dropout_10 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                4128
_________________________________________________________________
dropout_11 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,104,257
Trainable params: 104,257
Non-trainable params: 1,000,000
_________________________________________________________________
```

Figure: Proposed BiLSTM network architecture

- **2.5 Validation data:** The dataset is splited into two parts: training and testing dataset. Training data accounts for 80% of original data and the rest is used for testing. Validation data accounts for 30% of training data.
- **2.6 Evaluation metrics:** We used different evaluation metrics to evaluate our models, such as precision, recall, accuracy and f1-score. These metrics can be computed by using following equations:

$$\text{Accuracy} = ((TP+TN)/(TP+TN+FP+FN))*100$$

Recall = (TP/(TP+FN))*100

Precision = (TP/(TP+FP))*100

F1-score = ((2*precision*recall)/(precision+recall))*100

3. Implementation Details:

- Download the ISOT Fake News dataset from this link https://www.uvic.ca/engineering/ece/isot/datasets/fake-news/index.php . After extracting put the two files named 'Fake.csv' and 'True.csv' in 'data' folder. Load data from these two files separately in two dataframes using pandas and then merge them vertically into a single dataframe. Reshuffle this new dataframe to avoid any order. Combined 'title' and 'text' column into single 'text' column for training and testing.
- Removed HTML tags, punctuations and alphanumeric words from the text. Convert text to lowercase and then remove stopwords from it. Perform stemming with snowball stemmer. Split the data into training and testing. Training data will be 80% of original data and the rest will be testing data.
- Created A Tokenizer with Keras for Tokenizing The Words And Created Sequences Of Tokenized Words. Here we used top 10000 words for vocabulary size. Padded each sequence to a length of 300.
- Used 'glove.6B.100d.txt' file to get the pretrained weighted word vectors. Download 'glove.6B.100d.txt' file from this link https://drive.google.com/file/d/1QJxQUn_BD6-cg4BqYOKTl99r-DKj3LKS/view?usp=sharing and put this file into 'sources' folder.
- Train the proposed Bidirectional LSTM model on this data. Plot the training and validation accuracy as a function of epochs. Plot the training and validation loss as a function of epochs. Save the weights of the best model and then check the performance on test data using this best model.
- Download the best trained model named 'fake_news_model.h5' from this link https://drive.google.com/file/d/1MeQsYVZmtYtf5j93qtx6N-Z3KSKmPmtK/view?usp=sharing and put it in the folder named 'trained_model' which is inside the folder 'src'.
- Folder structure: project (main folder) →

(1). src (sub-folder) – It contains one source code file named 'Fake News Detection.ipynb'. It also contains one sub-folder named 'trained_model'. 'fake_news_model.h5' file should be put in 'trained_model' folder.

(2). data (sub-folder) – It contains one file named 'data.txt' which contains link for the dataset. Put 'Fake.csv' and 'True.csv' files in this folder.

(3). doc (sub-folder) – It contains one pdf file and its name is 'project report.pdf'.

(4). presentation (sub-folder) – It contains one pdf file and its name is 'project presentation.pdf'.

(5). sources (sub-folder) – It contains one pdf file named 'Fake News Detection using Deep Recurrent Neural Networks.pdf'. Put 'glove.6B.100d.txt' file into this folder.

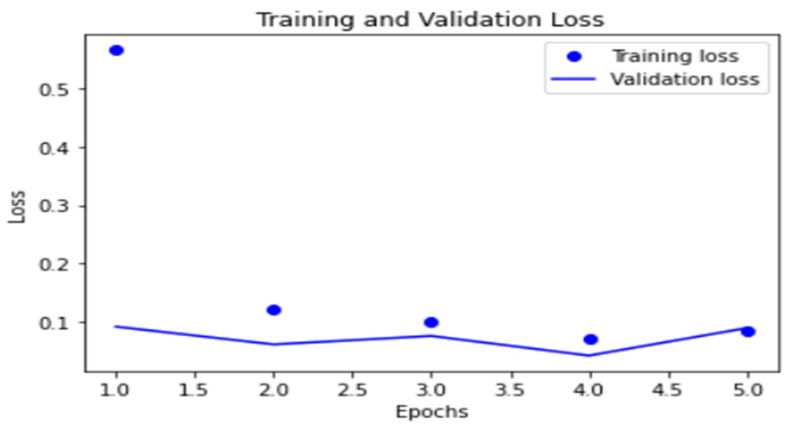
4. Results and discussion:

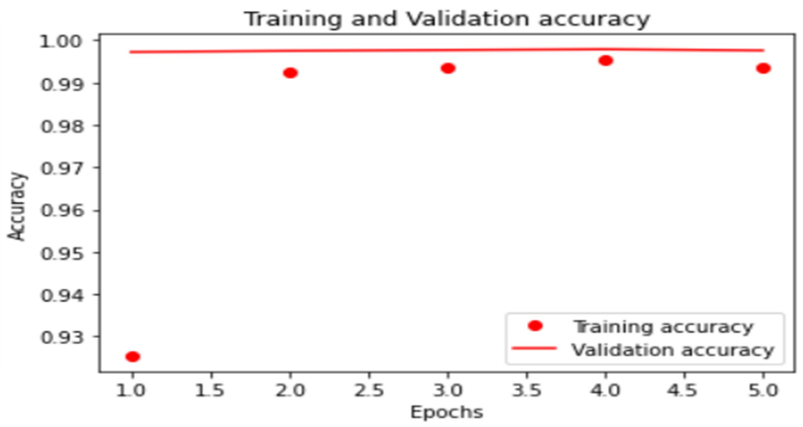
Table 1 – Model essential parameters:

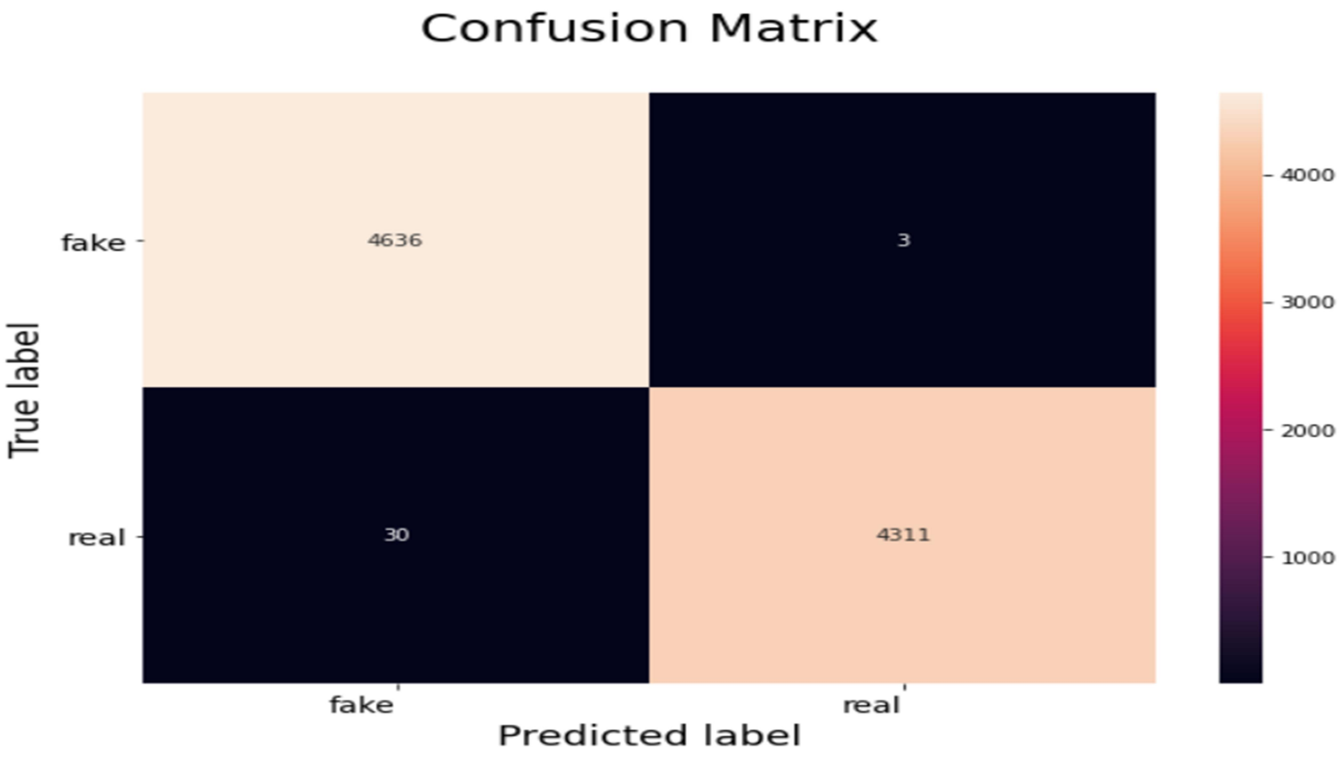
epochs	callbacks	optimizer	loss	Output activation function
5	ReduceLROnPlateau(monitor='val_acc', factor=0.5, patience=2, verbose = 1, min_lr=0.001)	Adam(lr=0.01)	binary_crossentropy	sigmoid

Table 2 – Model performance with multiple metrics on test data:

Loss	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
0.04746	99.632	99.93	99.30	99.618







From above results, we can clearly see that our model is of great quality and it can be used to detect fake news.

5. Future Work:

- We can try to combine other models like CNN, GRU with our model to see if integrated layers can improve the results. We will try to collect more data as data size is always the bottleneck of deep learning tasks. All news in our dataset is written in English, so we will also try to apply our model in non-English datasets.

6. References:

- https://paperswithcode.com/method/bilstm#
- https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4
- https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html
- https://nlp.stanford.edu/projects/glove/