CS416 Data Warehousing and Mining

Project - Final Report

Fighting Fake News on the COVID-19 Pandemic

Shruthan R (181CO250)
Videh Raj Nema (181CO258)

You can download the entire project from here (~900 MB). GloVe Vectors and saved models have been omitted in the submission made on Moodle due to file size.

1. Introduction

In this project, we have built a system which can identify if a given piece of news regarding the COVID-19 pandemic is true or false. Given the vast amount of fake news that spreads on social media platforms like WhatsApp and Twitter, by people with either insufficient knowledge or malicious intents, a system like this would be extremely useful to the society, to distinguish fact from fiction. An example of the kind of output we expect from the system is:

Statement: If you take Crocin thrice a day you are safe.
Output: False

Statement: Wearing a mask can protect you from the virus.
Output: True

2. Dataset

We have obtained the required data from a competition in Constraint@AAAI2021. The competition is hosted on codalab, and the dataset is provided on registration. (Click here to view the competition). Essentially, the dataset has tweets, which are classified as real or fake.

The dataset consists of 6420 training examples with three columns:

- id
- tweet
- label ("real"/"fake")

Additionally 2140 examples have been provided for validation and 2140 examples for testing. Hence 60% of the data is for training, 20% for validation and 20% for testing. Since the dataset is balanced, accuracy is a good metric to evaluate the models for this problem

3. Data Preprocessing

The following steps were carried out while preprocessing the dataset:

- Processing URLs by replacing "`http(\S)+`" and "`www(\S)+`" by empty space where `(\S)+` represents 1 or more non-whitespace characters.
- Replacing ampersand (&) with "and".
- Removing all non alphanumeric characters.
- Removing stop words, which would not help distinguishing between sentences/tweets.
- Converting labels to binary, if required

4. Frameworks/Libraries used:

Basic Libraries: Numpy, Pandas, Matplotlib, Collections.
For NLP tasks: NLTK and SpaCy
For Machine Learning tasks: Scikit-learn
For Deep Learning: PyTorch
For Transformers: The Hugging Face Transformers library

5. Classical Machine Learning Models

5.1 Classification Pipeline

The classification pipeline has three stages:

1. Converting the collection of tweets to a matrix of token counts using CountVectorizer.
2. Transforming the count-matrix to a normalized tf-idf representation. The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus
3. Passing the tf-idf representation to a classifier.

Three different classes of classifiers have been experimented, as described in the following sections.

5.2 Linear Models:

Logistic Regression and Support Vector Machines were the linear models used for classification. They both gave accuracies of 93%.

5.3 Tree Based Classifier

The Decision Trees classifier gave an accuracy of 85%

5.4 Ensemble Classifiers

Ensemble classifiers like Random Forests and Gradient Boosted Classifiers were used. They gave test accuracies of 91% and 87% respectively.

The results are summarized below:

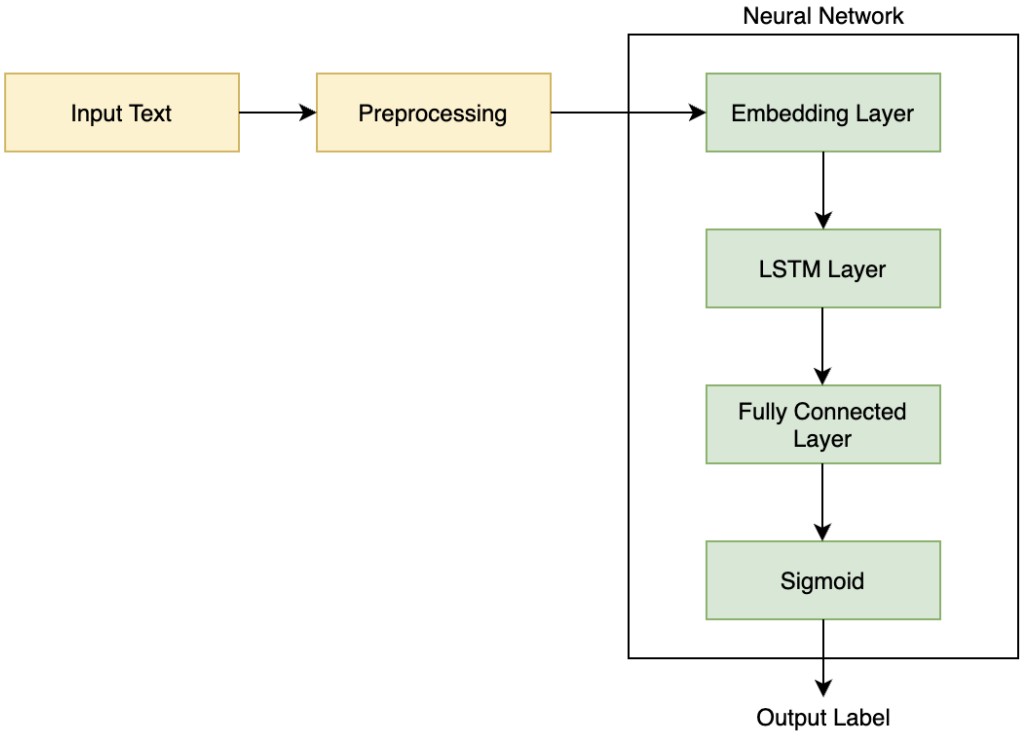
Classification Algorithm	Accuracy	F1 Score
Support Vector Machines	0.93	0.94
Logistic Regression	0.93	0.93
Random Forests	0.91	0.91
Gradient Boosting Classifier	0.87	0.87
Decision Trees	0.85	0.86

From these, we see that the linear classifiers performed the best, with giving a test-accuracy of 93%.

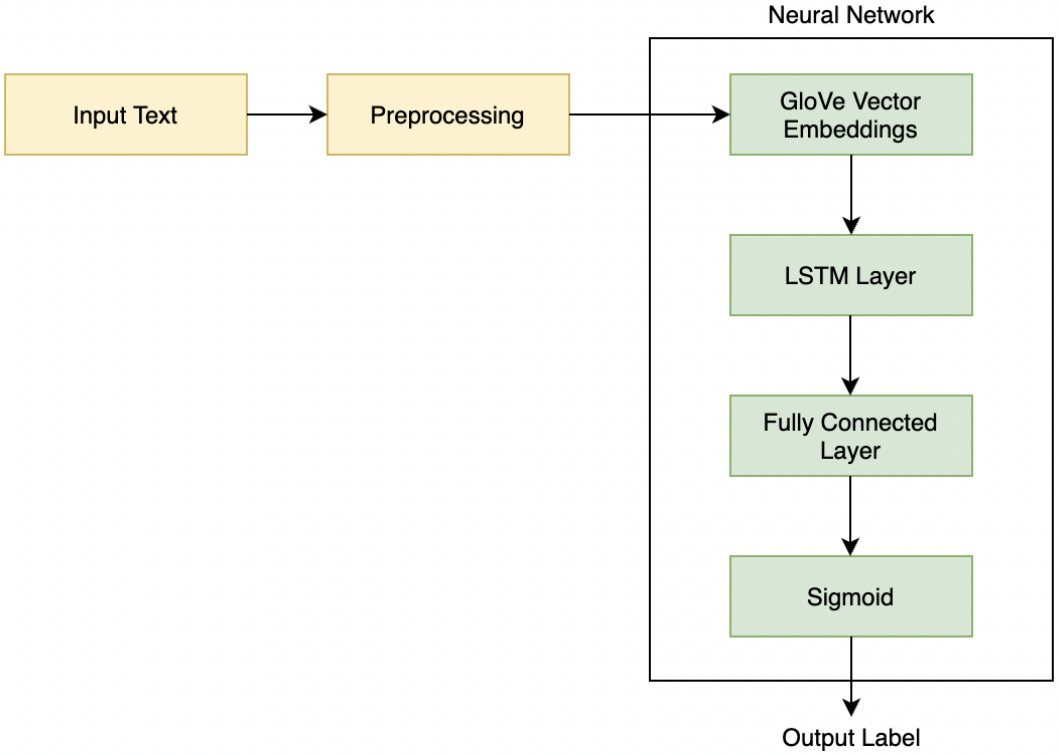
6. Deep Learning Models

The following models using neural networks have been implemented:

6.1 LSTM without GloVe



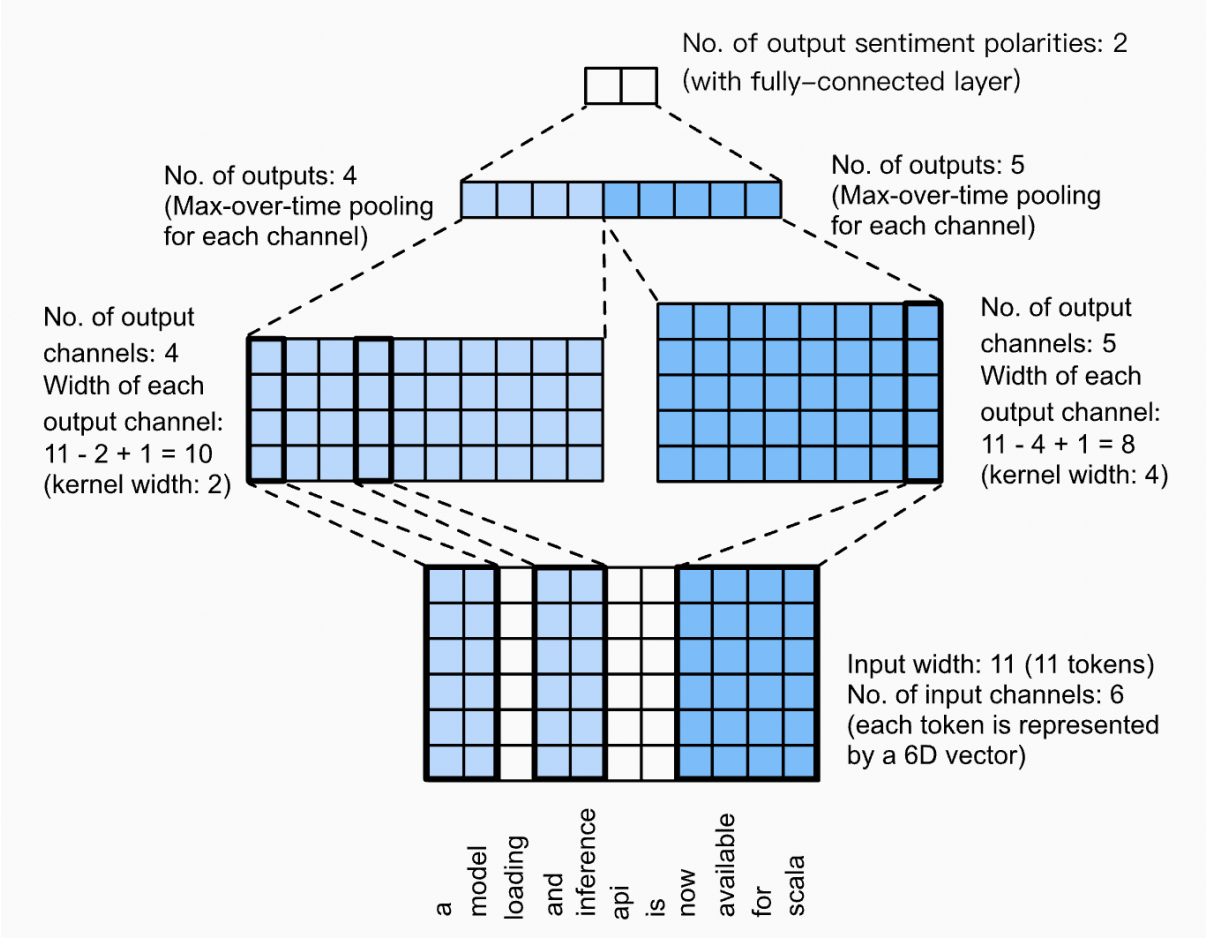
6.2 LSTM with Glove



While the LSTM without GloVe vectors model had a test accuracy of 90.79%, using GloVe vectors for the word embeddings increased the accuracy to 92.43%.

6.3 CNN (TextCNN)

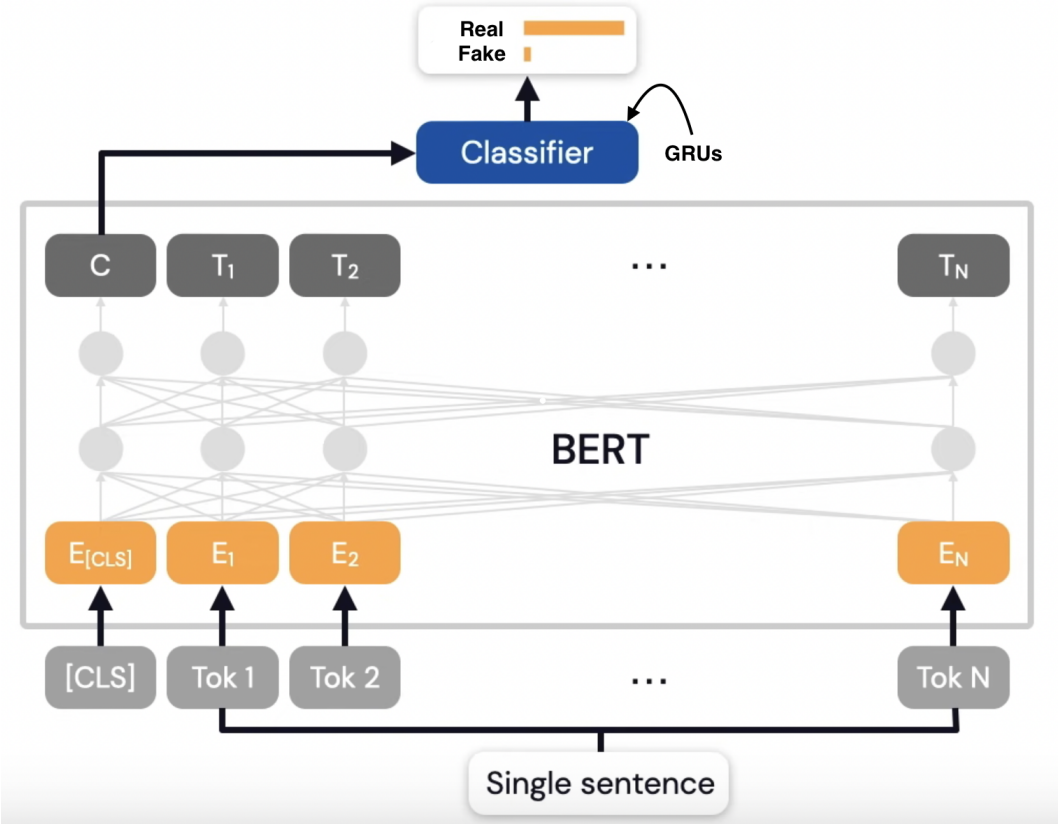
The preprocessing step remains the same, as in the previous two architectures. The neural network architecture however changes. The below is a representation of the architecture for a sentence with 11 tokens:



The final layer may equivalently be implemented as a softmax layer with two units or single sigmoid layer (as done in the code).

This model gave an test-accuracy of 93.04%, which is better than both versions of the LSTM model.

6.4 Transformer + GRU



In this model, BERT (Bidirectional Encoder Representations from Transformers) was used to encode/represent the individual sentences. The encoded sentences were then passed through a GRU-based classifier which culminated in a sigmoid layer. This model gave the best testing accuracy of 96.03%.

The results for the deep learning based models have been summarized below:

Model	Test Accuracy
LSTM without GloVe	90.79%
LSTM with GloVe	92.43%
CNN	93.04%
Transformer + GRU	96.03%

References:

1. https://scikit-learn.org/stable/user_guide.html
2. https://pytorch.org/docs/stable/index.html
3. https://d2l.ai/
4. https://www.youtube.com/watch?v=LE3NfEULV6k (A talk by a Google NLP researcher as a part of the Tensorflow's ML Talk Series)