Machine Learning Model to Classify one's Political Orientation

Data: Reddit Posts

Liberal VS Conservative

CIS 509

Assignment 1

By Sudhanshu Mishra

Cohort C

Table of Contents

Dataset:

Header row of the Dataset:

The dataset that is given in the JSON format contains a text column and a label, the label classifies it as liberal or conservative, this will be how we'll train our model.

	text	**Label**
0	50 Years On, The Feminist Press Is Radical and...	**Liberal**
1	Anti-worker bills are working their way through...	**Liberal**
2	The FBI Seized Almost $1 Million From This Fam...	**Conservative**
3	Stephanie Grisham's Book Details Trump's 'Terr...	**Liberal**
4	How Trump kept peace	**Conservative**

Vectorizing the Dataset:

```
if row['label'] == "Liberal":
  label = 1 # Liberal
else:
  label = 0 # Conservative
```

We will vectorize the dataset and set the liberal's as 1 and the conservatives as 0.

We will use **TFIDF Vectorizer** with max features of 10000

```
vectorizer = TfidfVectorizer(max_features=10000) # the value of max_f
vectorizer.fit(review_corpus) # fit the vectorizer with the entire da
# once we fitted the vectorizer, we use it for converting raw text (r
vectorized_review = vectorizer.transform(review_corpus)
```

```
vectorized_review
```

```
<5000x10000 sparse matrix of type '<class 'numpy.float64'>'
        with 58528 stored elements in Compressed Sparse Row format>
```

Converting Vectorized Data to NumPy Variables:

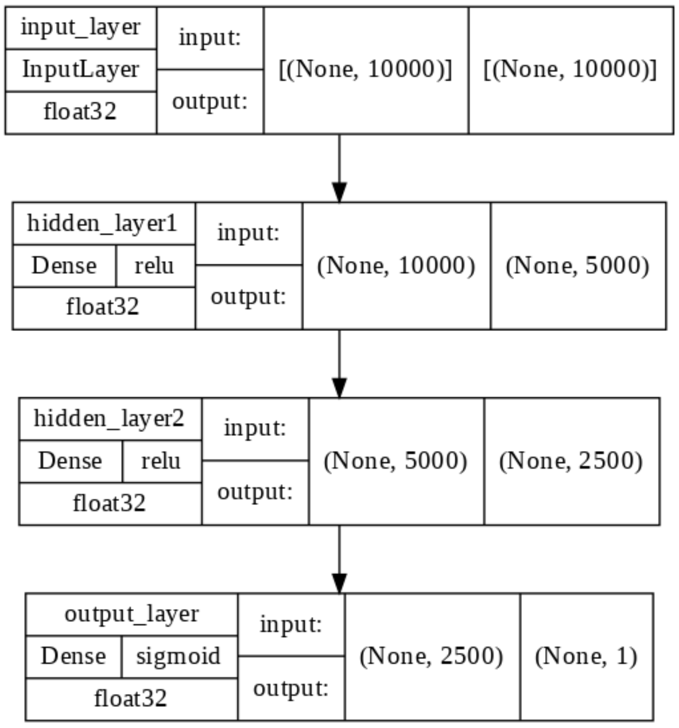
```
X, Y = vectorized_review.toarray(), np.asarray(y)
```

```
(5000, 10000)
<class 'scipy.sparse.csr.csr_matrix'>
```

It is important to convert the Vectorized data to a NumPy Array to feed into our TensorFlow Keras model.

Visualizing the Model Layers:

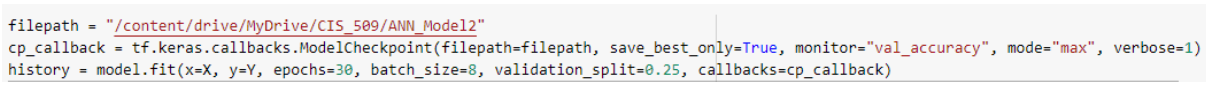
As seen below we will be using a 3-layer model with two hidden layers, the first hidden layer will have 5000 features and the 2nd will have 2500 features followed by the output layer. The activation functions used are Relu and Relu for hidden layers, and Sigmoid for the Output layer because we want a 1 or 0 output for this model, therefore, we use the sigmoid/logistic function.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 10000)]           0

 hidden_layer1 (Dense)       (None, 5000)              50005000

 hidden_layer2 (Dense)       (None, 2500)              12502500

 output_layer (Dense)        (None, 1)                 2501

=================================================================
Total params: 62,510,001
Trainable params: 62,510,001
Non-trainable params: 0
_________________________________________________________________
```

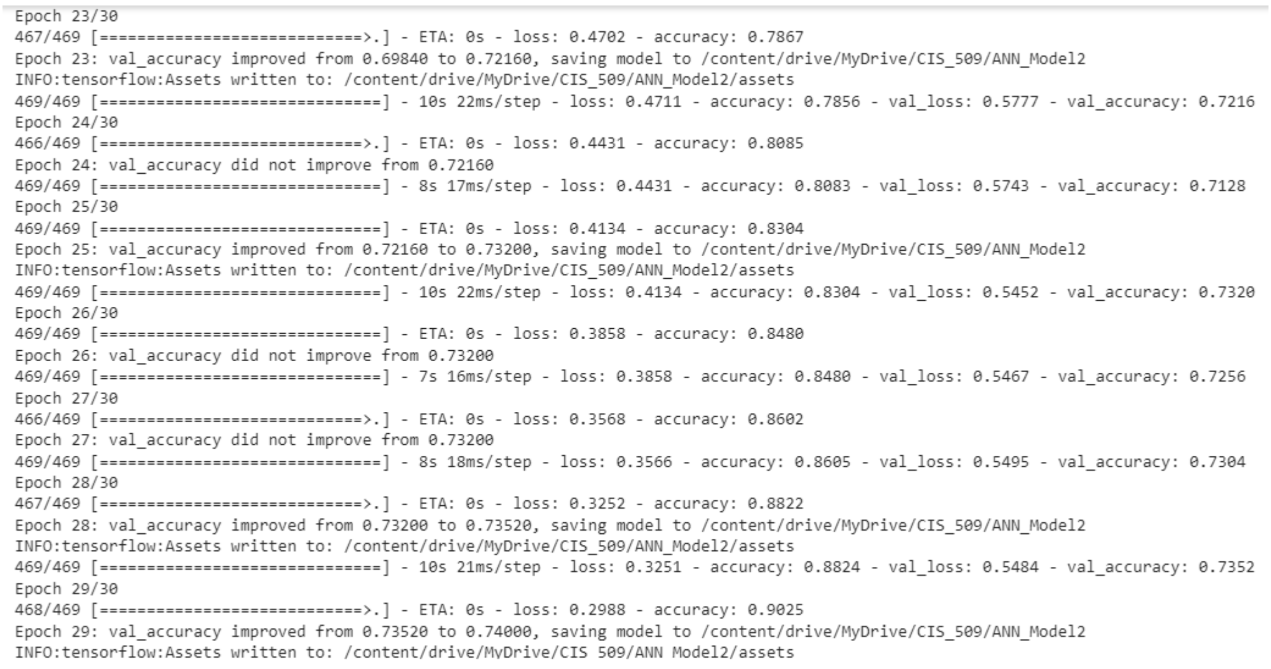
Fitting the Model:

We will fit and run the model using GPU runtime based in Google Colab. For this analysis we will save our best model using high validation accuracy, which will be justified by the following line of code.



```
filepath = "/content/drive/MyDrive/CIS_509/ANN_Model2"
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=filepath, save_best_only=True, monitor="val_accuracy", mode="max", verbose=1)
history = model.fit(x=X, y=Y, epochs=30, batch_size=8, validation_split=0.25, callbacks=cp_callback)
```

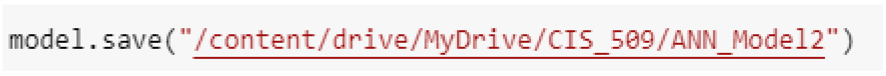
We will only run the model for 30 epochs with a batch size of 8 and a 75%-25% split. These can be changed depending on the requirements.



```
Epoch 23/30
467/469 [============================>.] - ETA: 0s - loss: 0.4702 - accuracy: 0.7867
Epoch 23: val_accuracy improved from 0.69840 to 0.72160, saving model to /content/drive/MyDrive/CIS_509/ANN_Model2
INFO:tensorflow:Assets written to: /content/drive/MyDrive/CIS_509/ANN_Model2/assets
469/469 [==============================] - 10s 22ms/step - loss: 0.4711 - accuracy: 0.7856 - val_loss: 0.5777 - val_accuracy: 0.7216
Epoch 24/30
466/469 [============================>.] - ETA: 0s - loss: 0.4431 - accuracy: 0.8085
Epoch 24: val_accuracy did not improve from 0.72160
469/469 [==============================] - 8s 17ms/step - loss: 0.4431 - accuracy: 0.8083 - val_loss: 0.5743 - val_accuracy: 0.7128
Epoch 25/30
469/469 [==============================] - ETA: 0s - loss: 0.4134 - accuracy: 0.8304
Epoch 25: val_accuracy improved from 0.72160 to 0.73200, saving model to /content/drive/MyDrive/CIS_509/ANN_Model2
INFO:tensorflow:Assets written to: /content/drive/MyDrive/CIS_509/ANN_Model2/assets
469/469 [==============================] - 10s 22ms/step - loss: 0.4134 - accuracy: 0.8304 - val_loss: 0.5452 - val_accuracy: 0.7320
Epoch 26/30
469/469 [==============================] - ETA: 0s - loss: 0.3858 - accuracy: 0.8480
Epoch 26: val_accuracy did not improve from 0.73200
469/469 [==============================] - 7s 16ms/step - loss: 0.3858 - accuracy: 0.8480 - val_loss: 0.5467 - val_accuracy: 0.7256
Epoch 27/30
466/469 [============================>.] - ETA: 0s - loss: 0.3568 - accuracy: 0.8602
Epoch 27: val_accuracy did not improve from 0.73200
469/469 [==============================] - 8s 18ms/step - loss: 0.3566 - accuracy: 0.8605 - val_loss: 0.5495 - val_accuracy: 0.7304
Epoch 28/30
467/469 [============================>.] - ETA: 0s - loss: 0.3252 - accuracy: 0.8822
Epoch 28: val_accuracy improved from 0.73200 to 0.73520, saving model to /content/drive/MyDrive/CIS_509/ANN_Model2
INFO:tensorflow:Assets written to: /content/drive/MyDrive/CIS_509/ANN_Model2/assets
469/469 [==============================] - 10s 21ms/step - loss: 0.3251 - accuracy: 0.8824 - val_loss: 0.5484 - val_accuracy: 0.7352
Epoch 29/30
468/469 [============================>.] - ETA: 0s - loss: 0.2988 - accuracy: 0.9025
Epoch 29: val_accuracy improved from 0.73520 to 0.74000, saving model to /content/drive/MyDrive/CIS_509/ANN_Model2
INFO:tensorflow:Assets written to: /content/drive/MyDrive/CIS_509/ANN_Model2/assets
```

We notice that Validation accuracy improves gradually.

Saving the Model:



```
model.save("/content/drive/MyDrive/CIS_509/ANN_Model2")
```

We save it using a simple model.save() command. Saving is important because we can retrieve it at a future point to test it on any new data points we get.

Testing the model on new data:

We introduce two texts to the model:

Data1:
"Anarchocapitalism, in my opinion, is a doctrinal system which, if e
ver implemented, would lead to   forms of tyranny and oppression tha
t have few counterparts in human history."

Data2:
"Biden's Response to Putin's Invasion of Ukraine Has Been His Finest
 Moment"

We will pass the new data as a list to our Tf Keras model that we have saved.

```
New_Data = ["Anarcho-
capitalism, in my opinion, is a doctrinal system which, if ever impleme
nted, would lead to forms of tyranny and oppression that have few count
erparts in human history.","Biden's Response to Putin's Invasion of Ukr
aine Has Been His Finest Moment"]
```

Loading the saved Model:

```
] #load your model here
  new_model = tf.keras.models.load_model("/content/drive/MyDrive/CIS_509/ANN_Model2")
```

We load the saved model from our drive storage location. Now this saved model can be used to predict any text passing through it as one of the two labels, liberal(1) or conservatives(0).

We will now Vectorize and Fit the model using the New Data:

```
vectorized_new_data = vectorizer.transform(New_Data)
new_x = vectorized_new_data.toarray()

new_model.predict(new_x)
```

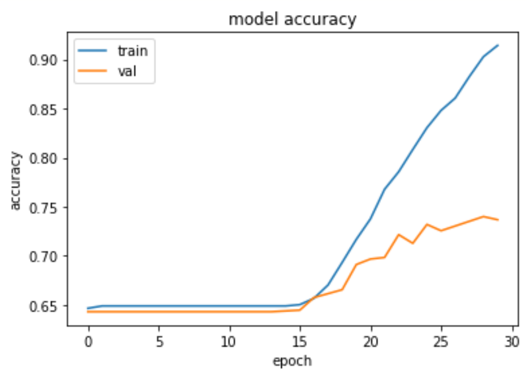
Results:

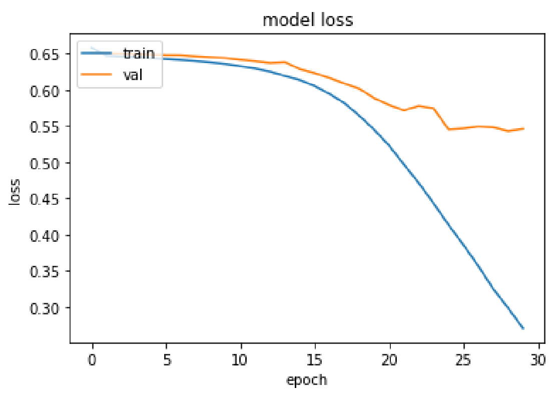
```
 array([[0.7673081 ],[0.04765433]], dtype=float32)
```

The results show that the first part of the new data is predicted as Liberal 0.767 and the 2nd New data is predicted as Conservative. Anything above 0.5 is labelled as liberal and below 0.5 would be labelled as conservative based on our model.

Conclusions and Visualizations:

Thus, this is how we create a model based on labelled text data. There are of course ways we can improve this model using POS tags as well as vectorizing the current text information. We can further visualize the model accuracy and loss as below:


model accuracy
train
val
accuracy
epoch


model loss
train
val
loss
epoch