DL-AS2-REPORT

Naveen Kumar Pathi 500187816

Venkatesh Policherla 500194692

Sai Chand Devarapalli 500192020

Krishna Vamsi Vanga 500187921

Dorai Charan Muthineni 500185125

Problem Statement: - In this project, we try to implement a Twitter sentiment analysis model that helps to detect the negative tweets.

Objective: -To use deep learning models like Simple-RNN on a NLP project to detect the negative tweets.

About Dataset: - The tweets have been pulled from Twitter and manual tagging has been done to them.

https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification?select=Corona_NLP_train.csv

The names and usernames have been given codes to avoid any privacy concerns.

```
[46]: train.head()
```

[46]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative

Data Cleaning: -

Here we keep only ‘Original Tweet’ and ‘Sentiment’ columns and remove the rest. In the sentiment column we replace ‘Negative’ and ‘Extremely Negative’ with 1 and the remaining to 0.

We see that there are no null values present in the dataset. Though our data is imbalanced we need no not make any changes as the deep learning models we use do not require any. Cleaned data looks as below.

```
[44]: train.head(5)
```

[44]:

	OriginalTweet	Sentiment
0	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	0
1	advice Talk to your neighbours family to excha...	0
2	Coronavirus Australia: Woolworths to give elde...	0
3	My food stock is not the only one which is emp...	0
4	Me, ready to go at supermarket during the #COV...	1

Tokenization: -

Tokenization is breaking the raw text into small chunks. Tokenization breaks the raw text into words, sentences called tokens.

In an RNN we input a sentence word by word. We represent every word as one hot vectors of dimensions: Numbers of words in Vocab +1.

What keras Tokenizer does is, it takes all the unique words in the corpus, forms a dictionary with words as keys and their number of occurrences as values, it then sorts the dictionary in descending order of counts. It then assigns the first value 1, second value 2 and so on. So, let's suppose word 'the' occurred the most in the corpus then it will assign index 1 and vector representing 'the' would be a one-hot vector with value 1 at position 1 and rest zeros.

```
# using keras tokenizer here
token = text.Tokenizer(num_words=None)
max_len = 1500

token.fit_on_texts(list(xtrain) + list(xvalid))
xtrain_seq = token.texts_to_sequences(xtrain)
xvalid_seq = token.texts_to_sequences(xvalid)

#zero pad the sequences
xtrain_pad = sequence.pad_sequences(xtrain_seq, maxlen=max_len)
xvalid_pad = sequence.pad_sequences(xvalid_seq, maxlen=max_len)

word_index = token.word_index
```

Model: Simple-RNN

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer.

```
%%time
with strategy.scope():
    # A simpleRNN without any pretrained embeddings and one dense layer
    model = Sequential()
    model.add(Embedding(len(word_index) + 1,
                        300,
                        input_length=max_len))
    model.add(SimpleRNN(100))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

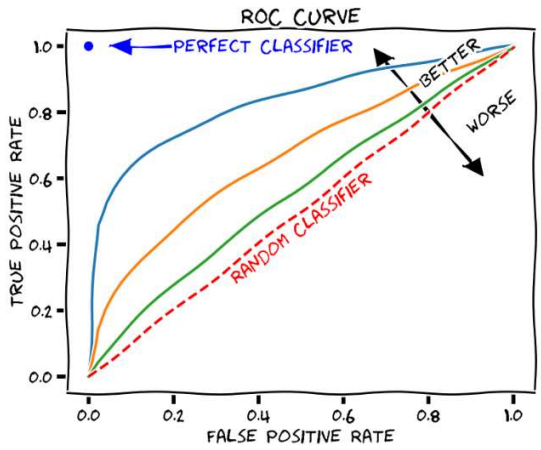
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 1500, 300)         25559700
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 100)               40100
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 25,599,901
Trainable params: 25,599,901
Non-trainable params: 0
_________________________________________________________________
Wall time: 434 ms
```

Fitting the model:

```
model.fit(xtrain_pad, ytrain, epochs=5, batch_size=64*strategy.num_replicas_in_sync) #Multiplying by Strategy to run on TPU's

Epoch 1/5
515/515 [==============================] - 479s 896ms/step - loss: 0.6374 - accuracy: 0.6438
Epoch 2/5
515/515 [==============================] - 518s 1s/step - loss: 0.2518 - accuracy: 0.8969
Epoch 3/5
515/515 [==============================] - 563s 1s/step - loss: 0.0418 - accuracy: 0.9873
Epoch 4/5
515/515 [==============================] - 725s 1s/step - loss: 0.0124 - accuracy: 0.9966
Epoch 5/5
515/515 [==============================] - 732s 1s/step - loss: 0.0053 - accuracy: 0.9987
<keras.callbacks.History at 0x273ff9cdd30>
```

AUC Score: Area Under Curve (AUC) score **represents the degree or measure of separability**. A model with higher AUC is better at predicting True Positives and True Negatives. AUC score measures the total area underneath the ROC curve. AUC is scale invariant and threshold invariant.



Final Observations: - We found AUC score for simple RNN model is 0.83.

```
[70]: scores = model.predict(xvalid_pad)
      print("Auc: {}".format(roc_auc(scores,yvalid)))

      Auc: 0.83
```

Conclusion: - Hence, we have successfully used a deep learning model- simple RNN on an NLP project with an AUC score of 0.83.