Application Exercise 2 - AISC2007 - Deep Learning 01

1. Dorai Charan Simha Muthineni - 500185125
2. Krishna Vamsi Vanga - 500187921
3. Naveen Kumar Pathi - 500187816
4. Sai Chand Devarapalli - 500192020
5. Venkatesh Policherla - 500194692

Configuring TPU

In [50]:
```
# Detect hardware, return appropriate distribution strategy
try:
    # TPU detection. No parameters necessary if TPU_NAME environment variable is
    # set: this is always the case on Kaggle.
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Running on TPU ', tpu.master())
except ValueError:
    tpu = None

if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
else:
    # Default distribution strategy in Tensorflow. Works on CPU and single GPU.
    strategy = tf.distribute.get_strategy()

print("REPLICAS: ", strategy.num_replicas_in_sync)
```

```
REPLICAS:  1
```

Importing required Libraries

In [56]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.models import Sequential
from keras.layers.recurrent import SimpleRNN
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.embeddings import Embedding
from sklearn import preprocessing, decomposition, model_selection, metrics, pipeline
from keras.preprocessing import sequence, text
```

Dataset

Collection of tweets from twitter related to Corona Virus and Covid-19

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

Data Preparation

In [57]:
```
train = pd.read_csv('Corona_NLP_train.csv')
test = pd.read_csv('Corona_NLP_test.csv')
```

In [58]:
```
train.head()
```

Out[58]:

	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

Here we require only tweets and and their sentiments so droping remaining columns

In [59]:
```
train.drop(['UserName','ScreenName','Location','TweetAt'],axis=1,inplace=True)
test.drop(['UserName','ScreenName','Location','TweetAt'],axis=1,inplace=True)
```

In [60]:
```
train.head(5)
```

Out[60]:

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

For easier use lets change the sentiment to either negative or non-negative

For negative we give 1 and for non-negative we give 0

In [61]:
```
train.replace(["Neutral", "Positive", "Extremely Positive"], 0, inplace=True)
test.replace(["Neutral", "Positive", "Extremely Positive"], 0, inplace=True)

train.replace(["Negative", "Extremely Negative"], 1, inplace=True)
test.replace(["Negative", "Extremely Negative"], 1, inplace=True)
```

In [62]:
```
train.head()
```

Out[62]:

	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

In [63]:
```
test.head()
```

Out[63]:

	OriginalTweet	Sentiment
0	TRENDING: New Yorkers encounter empty supermar...	1
1	When I couldn't find hand sanitizer at Fred Me...	0
2	Find out how you can protect yourself and love...	0
3	#Panic buying hits #NewYork City as anxious sh...	1
4	#toiletpaper #dunnypaper #coronavirus #coronav...	0

In [65]:
```
xtrain, xvalid, ytrain, yvalid = train_test_split(train.OriginalTweet.values,
train.Sentiment.values,
                                                  stratify=train.Sentiment.values,
                                                  random_state=42,
                                                  test_size=0.2, shuffle=True)
```

Function to Calculate 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.

In [66]:
```
def roc_auc(predictions,target):
    '''
    This methods returns the AUC Score when given the Predictions
    and Labels
    '''

    fpr, tpr, thresholds = metrics.roc_curve(target, predictions)
    roc_auc = metrics.auc(fpr, tpr)
    return roc_auc
```

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.

In [67]:
```
# 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.

In [68]:
```
%%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_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1500, 300)         25559700
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 100)               40100
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 25,599,901
Trainable params: 25,599,901
Non-trainable params: 0
_________________________________________________________________
Wall time: 234 ms
```

Model Fitting

In [69]:
```
# 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 [==============================] - 658s 1s/step - loss: 0.5852 - accuracy: 0.6945
Epoch 2/5
515/515 [==============================] - 680s 1s/step - loss: 0.6013 - accuracy: 0.6623
Epoch 3/5
515/515 [==============================] - 984s 2s/step - loss: 0.3105 - accuracy: 0.8599
Epoch 4/5
515/515 [==============================] - 1008s 2s/step - loss: 0.2235 - accuracy: 0.9066
Epoch 5/5
515/515 [==============================] - 728s 1s/step - loss: 0.1535 - accuracy: 0.9397
```

Out[69]:
```
<keras.callbacks.History at 0x27409b48550>
```

AUC Score of Simple-RNN model

In [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.