INTERNSHIP: PROJECT REPORT

Internship Project Title	Automate detection of different sentiments from textual comments and feedback
Project Title	Sentiment Analysis using Bi-LSTM Model
Name of the Company	TCS iON
Name of the Industry Mentor	Debashis Roy / Rushikesh
Name of the Institute	SRM Institute of Science and Technology, Chennai

Start Date	End Date	Total Effort (hrs.)	Project Environment	Tools used
28.05.2021	11.07.2021	25 hrs	Google Colab, Chrome, MacOs	Python 3

Project Synopsis:

Sentiment analysis is contextual mining of text which recognizes and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. With the popularity of social networks, and ecommerce websites, sentiment analysis has become a more active area of research in the past few years. On a high level, sentiment analysis tries to understand the public opinion about a specific
product or topic, or trends from reviews or tweets. Sentiment analysis plays an important role in better understanding person's opinion, and also extracting social/political trends.

Solution Approach:

Over the year various learning algorithms such as Bayesian, support vector machines (SVMs) and neural networks are used for various NLP tasks like language modeling, sentiment analysis, syntactic parsing, and machine translation. In this project, deep recurrent neural network using stacked bi-LSTM was implemented to perform the task of sentiment analysis. A stack of two bidirectional LSTM layers was used to form a deep RNN as previous studies show bi-LSTM have achieved accuracy higher than conventional LSTM. For this project, IMDB dataset was used which contains movie reviews along with their associated binary sentiment polarity labels. It contains 50,000 reviews split evenly into 25k train
and 25k test sets. The overall distribution of labels is balanced (25k positive and 25k negative). Reviews are preprocessed, and each review is encoded as a sequence of word indexes (integers). This allows for quick filtering operations. The data after being preprocessed is sent into Embedding layer which is further sent to bi-LSTM layers and the activation layer that predicts the output. The LSTM layers are followed by two dense layers having 'relu' and 'linear' as the activation function and a dropout of 0.5 between them.

Assumptions:

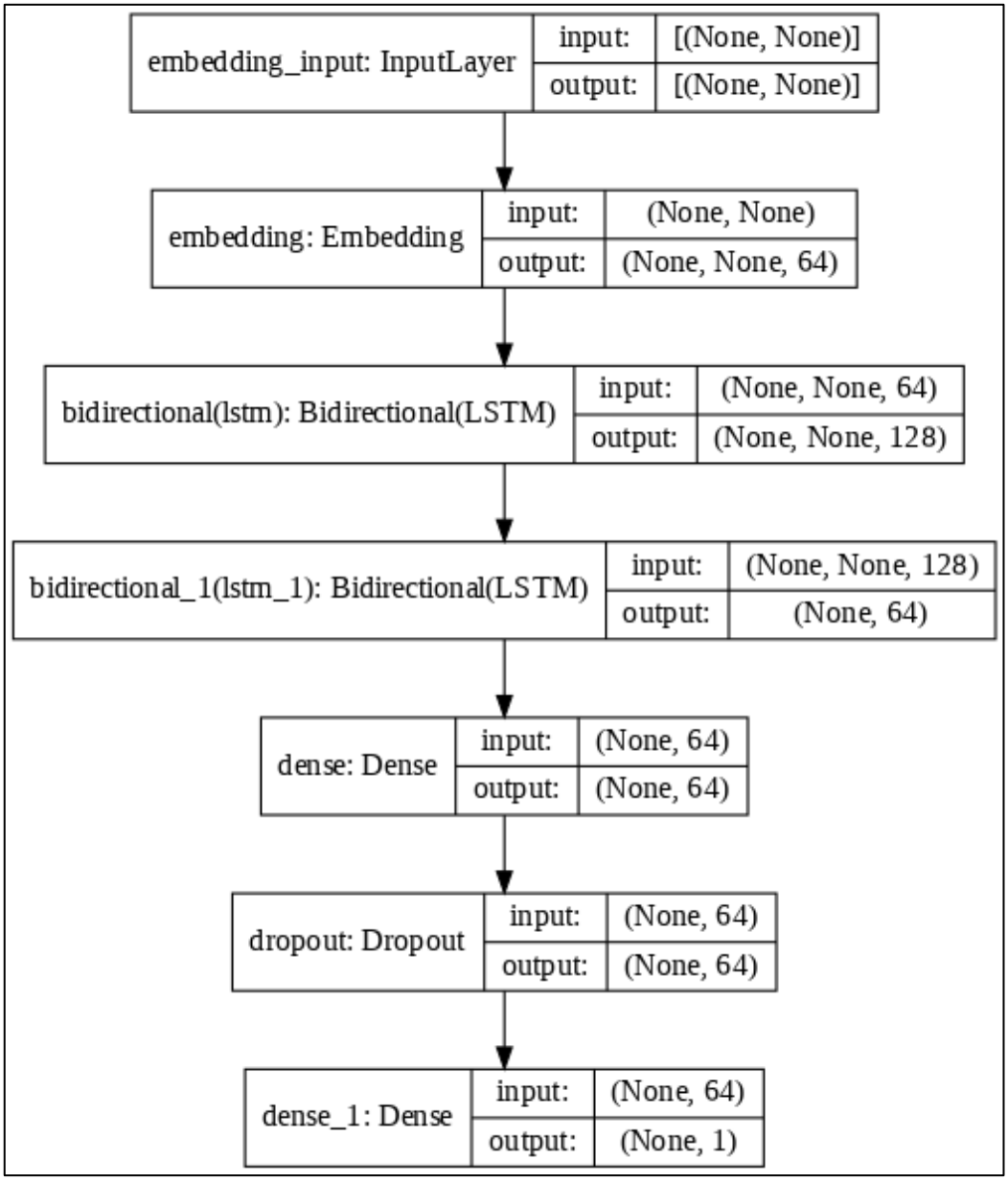
Reviews are already preprocessed in Keras IMDB dataset which is split in training and test sets by tensorflow_datasets library. This library also provides an encoder which encodes each review as a sequence of word indexes (integers).

Also, the training and testing sets are balanced, containing an equal number of positive and negative reviews. Thus, it is a binary or two-class-classification without 'neutral' label. But this model does not mask the padding applied to the sequences and predict the sentiments giving a prediction score i.e.

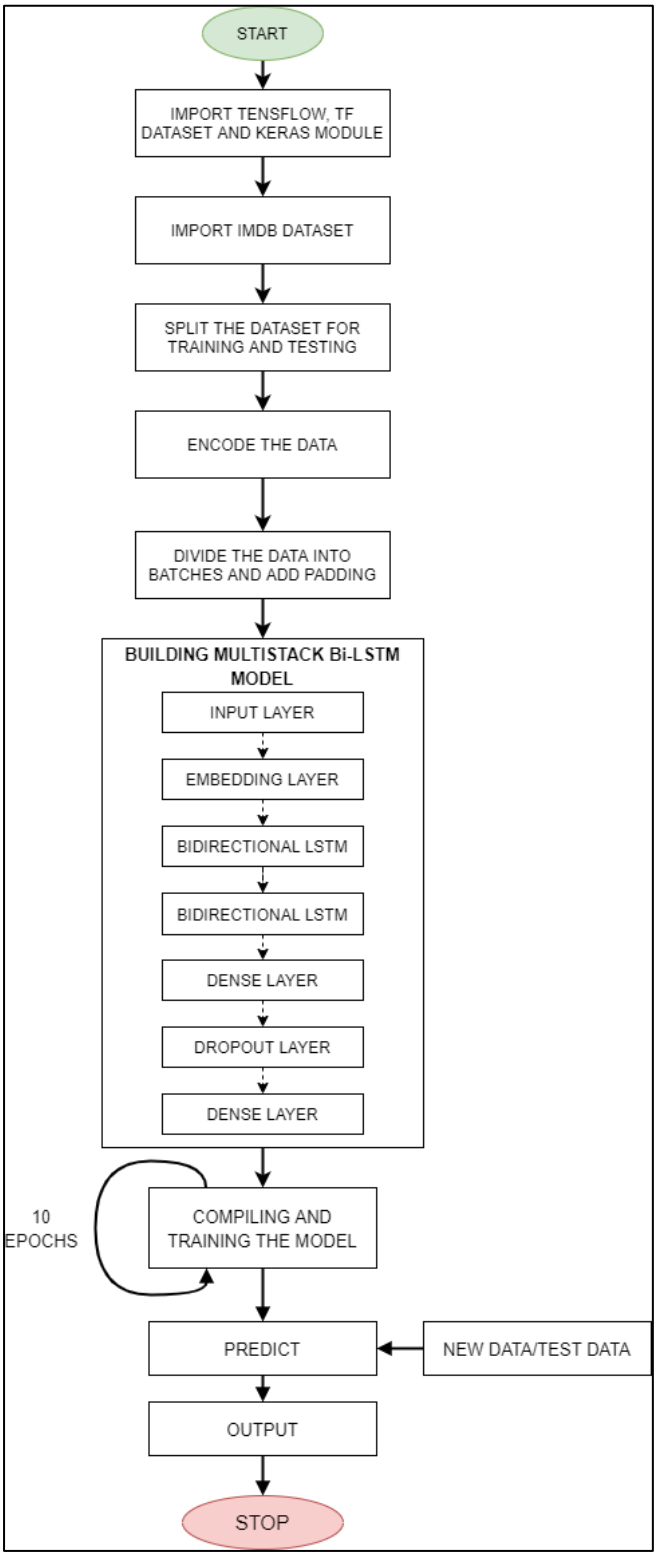
- If the prediction is >= 0.5, it is *positive*
- If the prediction is <= -1, it is *negative*
- If the prediction is between 0.5 and -1, it is *neutral*

Project Diagrams:

MODEL DIAGRAM



PROJECT FLOWCHART



Algorithms:
Recurrent Neural Network (RNN) using stacked Bidirectional Long Short-Term Memory (LSTM).

DETAILED WORKING OF THE PROJECT:

1. **Dataset:** IMDB dataset of 25,000 movies reviews is used for training and same for testing from IMDB, each labeled by sentiment (positive/negative).

2. **Data Preprocessing**: Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers). These encoded reviews are then divided into batchs and since the reviews are all of different lengths, thus padded_batch is used to zero pad the sequences while batching. Each batch has a shape of (batch_size, sequence_length) because the padding is dynamic each batch will have a different length. Thus, we initialize BUFFER_SIZE = 10000 and BATCH_SIZE = 64. The training set is shuffled on every time we run the cell causing the final accuracy to be varied every time.

3. **Model**: In this project, *tf.keras.Sequential* model is used as the all the layers in the model only have single input and produce single output and there are 6 layers
    i) **EMBEDDING LAYER:** The preprocessed reviews are first sent to the embedding layer. This layer turns positive indices into dense vectors of fixed size. These vectors are trainable. After training (on enough data), words with similar meanings often have similar vectors.

    ii) **BIDIRECTIONAL LSTM LAYER:** The LSTM consists of units or memory blocks in the recurrent hidden layer, which contains memory cells with self-connections storing the temporal state of the network. In addition to this the network has special multiplicative units called gates to control the flow of information in the network. The principle of Bidirectional LSTM is to split the neurons of a regular LSTM into two directions, one for positive time direction forward states, and another for negative time direction backward states, the output is not connected to inputs of the opposite direction states. By using two-time directions, input information from the past and future of the current time frame can be used unlike standard LSTM which requires the delays for including future information. In this model, a deep bi-LSTM structure stacked with two LSTM layers is used.

    iii) **DENSE LAYER:** The LSTM layers are followed by two dense layers having the relu and default linear activation function and dropout layer between them. Dense layer is the regular deeply connected neural network layer. The relu will output the input directly if it is positive, otherwise, it will output zero. The last layer is densely connected with a single output node. This uses the default linear activation function that outputs logit for numerical stability.

    iv) **DROPOUT LAYER:** The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

4. **Compiling and Training:** For this model, binary cross entropy loss function and an Adam optimizer was used for training. Since this is a binary classification problem, binary cross entropy loss function is better suited as it compares each of the predicted probabilities to actual class output which can be either 0 or 1. Cross Entropy is definitely a good loss function for Classification Problems, because it minimizes the distance between two probability distributions - predicted and actual. The Adam optimizer is used as it

combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. The learning rate hyperparameter is set to 0.0001. And then the model is traned over 10 epochs.

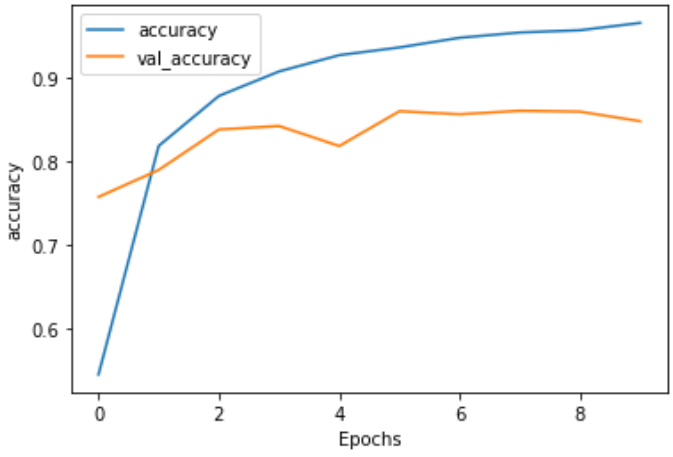
```
Epoch 1/10
391/391 [==============================] - 1650s 4s/step - loss: 0.6590 - accuracy: 0.5448 - val_loss: 0.5360 - val_accuracy: 0.7573
Epoch 2/10
391/391 [==============================] - 1616s 4s/step - loss: 0.4123 - accuracy: 0.8184 - val_loss: 0.4111 - val_accuracy: 0.7896
Epoch 3/10
391/391 [==============================] - 1618s 4s/step - loss: 0.3161 - accuracy: 0.8782 - val_loss: 0.3561 - val_accuracy: 0.8380
Epoch 4/10
391/391 [==============================] - 1620s 4s/step - loss: 0.2527 - accuracy: 0.9074 - val_loss: 0.3489 - val_accuracy: 0.8422
Epoch 5/10
391/391 [==============================] - 1633s 4s/step - loss: 0.2150 - accuracy: 0.9270 - val_loss: 0.4396 - val_accuracy: 0.8182
Epoch 6/10
391/391 [==============================] - 1614s 4s/step - loss: 0.1914 - accuracy: 0.9362 - val_loss: 0.3579 - val_accuracy: 0.8599
Epoch 7/10
391/391 [==============================] - 1609s 4s/step - loss: 0.1661 - accuracy: 0.9478 - val_loss: 0.3713 - val_accuracy: 0.8562
Epoch 8/10
391/391 [==============================] - 1637s 4s/step - loss: 0.1532 - accuracy: 0.9541 - val_loss: 0.4116 - val_accuracy: 0.8604
Epoch 9/10
391/391 [==============================] - 1641s 4s/step - loss: 0.1465 - accuracy: 0.9568 - val_loss: 0.4458 - val_accuracy: 0.8594
Epoch 10/10
391/391 [==============================] - 1634s 4s/step - loss: 0.1249 - accuracy: 0.9656 - val_loss: 0.4556 - val_accuracy: 0.8479
```

5. **Testing:** Testing is performed on the test dataset to obtain the following accuracy and loss

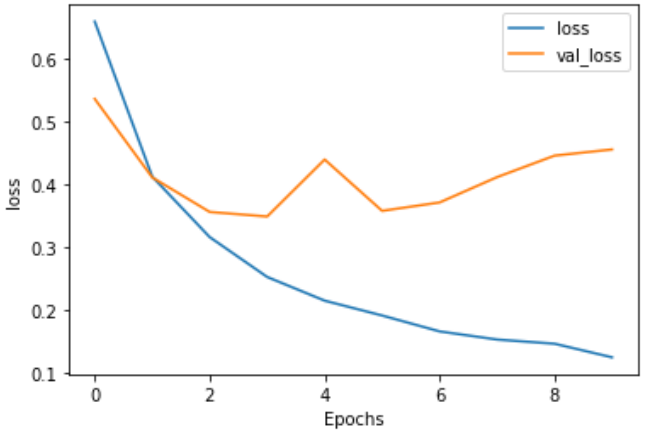
```
391/391 [==============================] - 351s 898ms/step - loss: 0.4689 - accuracy: 0.8438
Test Loss: 0.4688892662525177
Test Accuracy: 0.8437600135803223
```

GRAPHS

ACCURACY GRAPH



LOSS GRAPH



Outcome:

SAMPLE REVIEWS PREDICTION OUTPUT

```
[ ]  sample_pred_text = ('This movie was so so. The acting was medicore. Kind of recommend')
     predictions = sample_predict(sample_pred_text, pad = False)
     print(predictions)

     Prediction Score:  [[0.18905787]]
     NEUTRAL
```

```
[ ]  sample_pred_text = ('Absolutely terrible writing and dragged-out unnecessary dialogue')

     predictions = sample_predict(sample_pred_text, pad = False)
     print(predictions)

     Prediction Score:  [[-1.5297097]]
     NEGATIVE
```

```
     sample_pred_text = ("Loved the movie. brilliant")

     predictions = sample_predict(sample_pred_text, pad = False)
     print(predictions)

     Prediction Score:  [[0.50894576]]
     POSITIVE
```

Thus, a model consisting of Bidirectional Deep LSTM is presented for sentiment analysis of movie review. With the increase of complexity of the network the validation set accuracy increased and there by validation loss dropped. But increase in complexity clearly also increases the computational cost of the network.

Exceptions considered:

Since the dataset is shuffled every time the cell is run, small variations in accuracy and loss functions can be observed. This might give minor variance in the prediction values which may or may not affect the neutral sentiments.

Also, the final predicted value might slightly differ according to padding value in case of shorter sentences. *pad= False* gives better prediction value compared to *pad* = True in case of short senstences/paragraphs.

Enhancement Scope:

Future works might be done on optimization of the Bidirectional Deep LSTM for better increase in performance of the system. Different variations on the model like changing the activation function, varying the number of stacked LSTM layers, varying the input and output dimensions of the network might also increase the accuracy of the system.

Link to Code and executable file:

https://colab.research.google.com/drive/12oKTRoelQUXzPpJLE4IyclBLZW4VPdvS?usp=sharing