SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

COURSE CODE: DJ19DSC501 **DATE:**

COURSE NAME: Machine Learning - II **CLASS: AY 2023-24**

LAB EXPERIMENT NO.6

60009210105

Amitesh Sawarkar

D 12

AIM / OBJECTIVE:

Implement LSTM Sentiment Analysis on text dataset to evaluate customer reviews.

DESCRIPTION OF EXPERIMENT:

Python sentiment analysis is a methodology for analyzing a piece of text to discover the sentiment hidden within it. It accomplishes this by combining machine learning and natural language processing (NLP). Sentiment analysis allows you to examine the feelings expressed in a piece of text. It is essential for businesses to gauge customer response.

Preprocessing -

1) Normalization - Words which look different due to casing or written another way but are the same in meaning need to be process correctly. Normalisation processes ensure that these words are treated equally. For example, changing numbers to their word equivalents or converting the casing of all the text.
    a) Casing the Characters - Converting character to the same case so the same words are recognised as the same. (all lowercase)
    b) Removing - Stand alone punctuations, special characters and numerical tokens are removed as they do not contribute to sentiment which leaves only alphabetic characters. This step needs the use of tokenized words as they have been split appropriately for us to remove. We need to remove the special characters, numbers from the text. We can use the regular expression operations library of Python.

2) Tokenization - Tokenization is the process of breaking down chunks of text into smaller pieces. It converts text into tokens before transforming it into vectors. It is also easier to filter out unnecessary tokens. spaCy comes with a default processing pipeline that begins with tokenization, making this process a snap. In spaCy, you can do either sentence tokenization or word tokenization:

- Word tokenization breaks text down into individual words.
- Sentence tokenization breaks text down into individual sentences.

3) Stopwords - Stop words are the most commonly occuring words which are not relevant in the context of the data and do not contribute any deeper meaning to the phrase. In this case it contains no sentiment. We need to remove them as part of text preprocessing. nltk has a list of stopwords of every language.

4) Obtaining the stem words

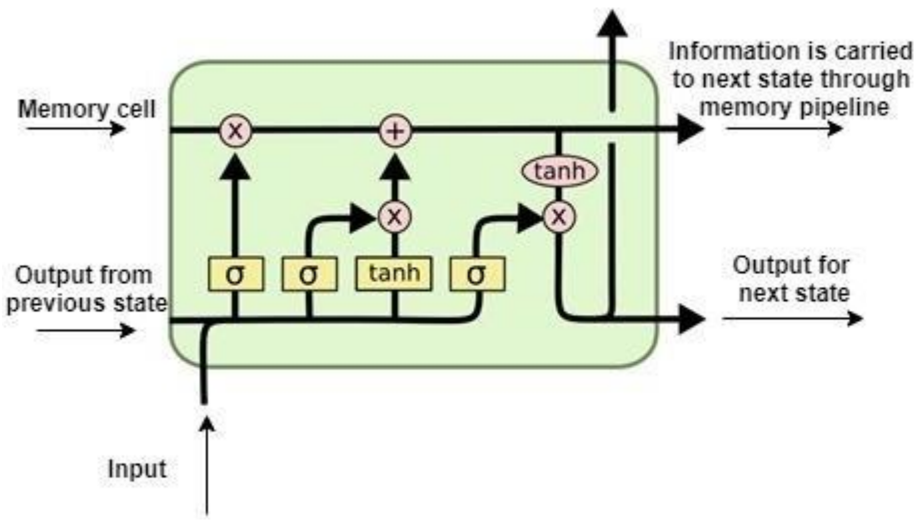
A stem is a part of a word responsible for its lexical meaning. The two popular techniques of obtaining the root/stem words are Stemming and Lemmatization

a) Stemming - Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words eating, eats, eaten is eat.

b) Lemmitization - This process finds the base or dictionary form of the word known as the lemma. This is done through the use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations)

5) Vectorization - use a count vectorizer from the Scikit-learn library to transform the text in data frame into a bag of words model, which will contain a sparse matrix of integers. The number of occurrences of each word will be counted.

Sentiment Analysis using LSTM: Use Keras



Hyperparameters to tune -

1. Layers - Explore additional hierarchical learning capacity by adding more layers and varied numbers of neurons in each layer
2. Number of inputs in dense layer - Dense layers improve overall accuracy and 5–10 units or nodes per layer is a good base
3. Dropout - Slow down learning with regularization methods like dropout on the recurrent LSTM connections.A good starting point is 20% but the dropout value should be kept small (up to 50%). The 20% value is widely accepted as the best compromise between preventing model overfitting and retaining model accuracy.
4. Learning Rate - This hyperparameter defines how quickly the network updates its parameters.
5. Decay Rate - weight decay can be added in the weight update rule that makes the weights decay to zero exponentially, if no other weight update is scheduled. After each update, the weights are multiplied by a factor slightly less than 1, thereby preventing them from growing to huge. This specifies regularization in the network.

6. Number of epochs

Sentiment Analysis using TextBlob:

TextBlob is a Python library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

The two measures that are used to analyze the sentiment are:

- Polarity – talks about how positive or negative the opinion is
- Subjectivity – talks about how subjective the opinion is

TextBlob(text).sentiment gives us the Polarity, Subjectivity values.

Polarity ranges from -1 to 1 (1 is more positive, 0 is neutral, -1 is more negative)

Subjectivity ranges from 0 to 1(0 being very objective and 1 being very subjective)

```
res = TextBlob("I love horror films").sentiment
res
```

```
Sentiment(polarity=0.5, subjectivity=0.6)
```

Example of TextBlob sentiment

Workflow -

1. Preprocess data.
2. Split data into training and evaluation sets.
3. Select a model architecture.
4. Use training data to train model.
5. Use test data to evaluate the performance of model.

1. Apply preprocessing techniques and LSTM on the dataset. Show accuracy achieved on the test dataset by providing classification report.

2. Perform LSTM hyperparameter tuning to improve accuracy score.
3. Show how LSTM model compares to built-in classifier provided by TextBlob.
4. State the applications of sentiment analysis
5. State the challenges faced while performing sentiment analysis.

Long Short Term Memory (LSTM)

Importing necessary packages/libraries

```
import numpy
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.datasets import imdb
from keras.models import Sequential
from keras.preprocessing import sequence
from keras.layers import Dense, LSTM, Embedding
from keras.preprocessing.sequence import pad_sequences
```

Importing and arranging data

```
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```

```
print(len(X_train[1]))
```

```
189
```

Making each review of uniform length, 600 characters

```
max_review_length = 600
X_train = pad_sequences(X_train, maxlen=max_review_length)
X_test = pad_sequences(X_test, maxlen=max_review_length)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(25000, 600) (25000,)
(25000, 600) (25000,)
```

Creating the model

```
embedding_vector_length = 64
```

```
model = Sequential([
    Embedding(top_words + 1, embedding_vector_length,
              input_length=max_review_length),
    LSTM(500),
    Dense(50, activation='relu'),
    Dense(100, activation='relu'),
    Dense(100, activation='relu'),
    Dense(50, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 600, 64)           320064

 lstm_1 (LSTM)               (None, 500)               1130000

 dense_5 (Dense)             (None, 50)                25050

 dense_6 (Dense)             (None, 100)               5100

 dense_7 (Dense)             (None, 100)               10100

 dense_8 (Dense)             (None, 50)                5050

 dense_9 (Dense)             (None, 1)                 51

=================================================================
Total params: 1495415 (5.70 MB)
Trainable params: 1495415 (5.70 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
hist = model.fit(X_train, y_train, epochs=10, verbose=1, validation_data=(X_test, y_test))
```

```
Epoch 1/10
782/782 [==============================] - 121s 149ms/step - loss: 0.6257 - accuracy: 0.6315 - val_loss: 0.6891 - val_accuracy: 0.5694
Epoch 2/10
782/782 [==============================] - 112s 143ms/step - loss: 0.5343 - accuracy: 0.7274 - val_loss: 0.3989 - val_accuracy: 0.8317
Epoch 3/10
782/782 [==============================] - 109s 139ms/step - loss: 0.3400 - accuracy: 0.8559 - val_loss: 0.3105 - val_accuracy: 0.8689
Epoch 4/10
782/782 [==============================] - 107s 137ms/step - loss: 0.2474 - accuracy: 0.9002 - val_loss: 0.2818 - val_accuracy: 0.8842
Epoch 5/10
782/782 [==============================] - 107s 136ms/step - loss: 0.1956 - accuracy: 0.9226 - val_loss: 0.3199 - val_accuracy: 0.8752
Epoch 6/10
782/782 [==============================] - 106s 136ms/step - loss: 0.1545 - accuracy: 0.9406 - val_loss: 0.3009 - val_accuracy: 0.8804
Epoch 7/10
782/782 [==============================] - 105s 134ms/step - loss: 0.1269 - accuracy: 0.9539 - val_loss: 0.3785 - val_accuracy: 0.8711
Epoch 8/10
782/782 [==============================] - 105s 135ms/step - loss: 0.1016 - accuracy: 0.9640 - val_loss: 0.3546 - val_accuracy: 0.8782
Epoch 9/10
782/782 [==============================] - 88s 113ms/step - loss: 0.0819 - accuracy: 0.9707 - val_loss: 0.4112 - val_accuracy: 0.8767
Epoch 10/10
782/782 [==============================] - 105s 135ms/step - loss: 0.0618 - accuracy: 0.9796 - val_loss: 0.5317 - val_accuracy: 0.8744
```

Model Evaluation

```
scores = model.evaluate(X_test, y_test, verbose=0)
```

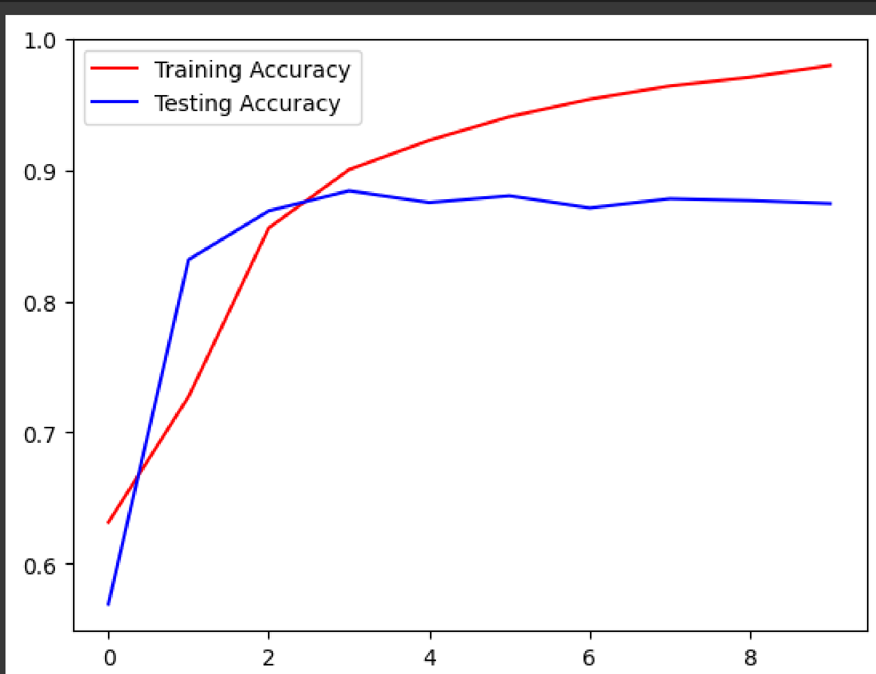
Validation Accuracy:

```
print(scores[1])
```

```
0.8744400143623352
```

Accuracy v/s Epochs

```
plt.plot(hist.history['accuracy'], color='r')
plt.plot(hist.history['val_accuracy'], color='b')
plt.legend(['Training Accuracy', 'Testing Accuracy'])
plt.show()
```



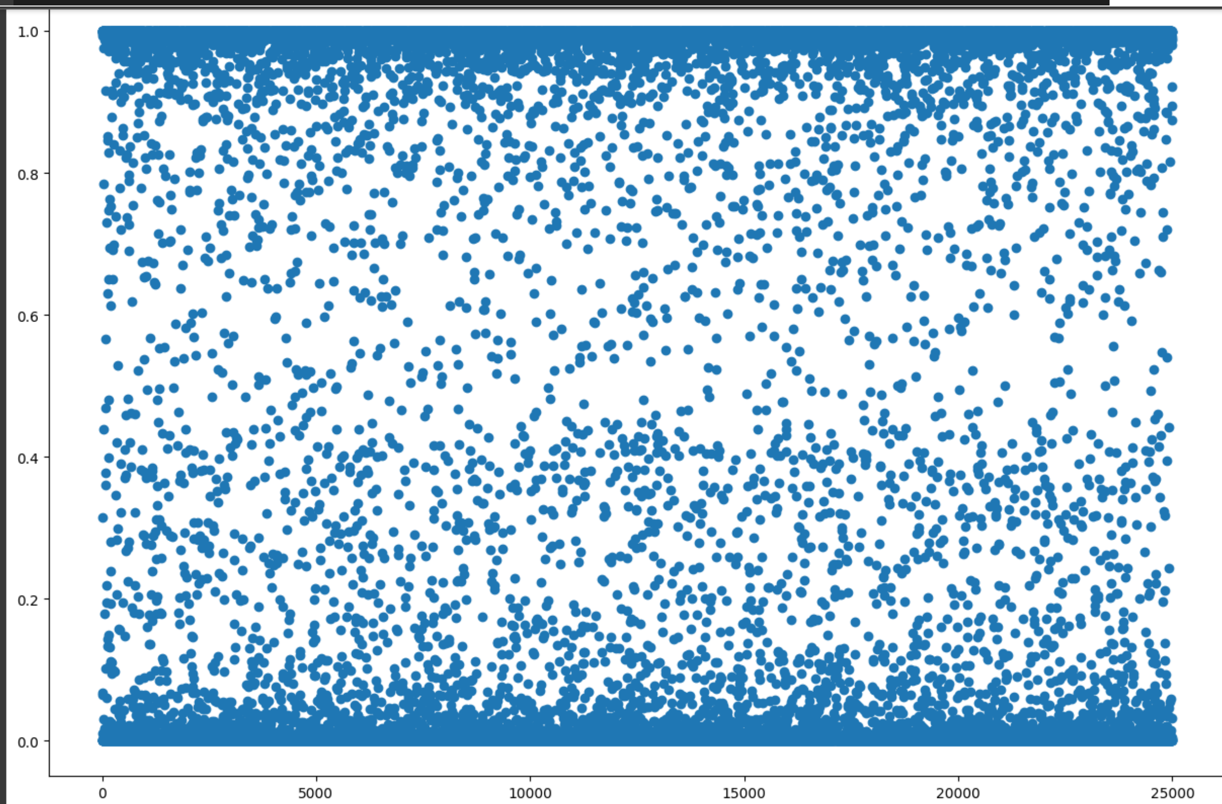
Prediction

```
y_pred = model.predict(X_test)
print(len(y_pred))
```

```
782/782 [==============================] - 24s 30ms/step
25000
```

Prediction plot

```
plt.figure(figsize=(15, 10))
plt.scatter(range(len(y_pred)), y_pred)
plt.show()
```



SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

As it can be seen, most of the predictions lie at either 0 or 1, we can apply further thresholding to finalize the categories

Thresholding for 2 final categories

```
for i in range(len(y_pred)):
    if (y_pred[i] >= 0.5):
        y_pred[i] = int(1)
    else:
        y_pred[i] = int(0)
```

Final evaluation

```
incorr = 0
for i in range(len(y_pred)):
    if y_pred[i] != y_test[i]:
        incorr += 1

print(incorr, '/', len(y_pred))
```

```
3139 / 25000
```

```
confuse_mat = tf.math.confusion_matrix(labels=y_pred, predictions=y_test)
plt.figure(figsize=(15, 8))
sns.heatmap(confuse_mat, annot=True, fmt='d')
plt.xlabel('Predicted Value')
plt.ylabel('True Value')
plt.show()
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)