Deep Learning and Neural Networks - Industry Assignment 2

Text Classification using Neural Networks - Classify Amazon Reviews using CNN and LSTM

Importing Libraries

In [1]:

```
import bz2
import nltk
import regex as re
import pandas as pd
import numpy as np
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

from keras.preprocessing import text,sequence
from keras.callbacks import EarlyStopping
from keras.layers import Dense,Embedding,LSTM,Dropout,SpatialDropout1D,GlobalMaxPooling1D, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.optimizers import SGD, Adam

from nltk.tokenize import TreebankWordTokenizer
from nltk.stem.regexp import RegexpStemmer

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
```

Loading the Dataset

In [2]:

```
def get_labels_and_texts(file):
    labels = []
    texts = []

    for line in bz2.BZ2File(file):
        x = line.decode("utf-8")
        labels.append(int(x[9])-1)
        texts.append(x[10:].strip())

    labels = labels[:int(len(labels)*0.01)]
    return np.array(labels), texts
```

Reading the Dataset

In [3]:

```
train_labels, train_texts = get_labels_and_texts('D:\\Khushi MCA\\MCA Semester 3\\Deep Learning and Neural Netw

test_labels, test_texts = get_labels_and_texts('D:\\Khushi MCA\\MCA Semester 3\\Deep Learning and Neural Networ

train_df=pd.DataFrame(zip(train_texts,train_labels),columns=['text','label'])

print(train_df.head())

test_df=pd.DataFrame(zip(test_texts,test_labels),columns=['text','label'])

print(test_df.head())
```

```
                                                text  label
0  Stuning even for the non-gamer: This sound tra...      1
1  The best soundtrack ever to anything.: I'm rea...      1
2  Amazing!: This soundtrack is my favorite music...      1
3  Excellent Soundtrack: I truly like this soundt...      1
4  Remember, Pull Your Jaw Off The Floor After He...      1
                                                text  label
0  Great CD: My lovely Pat has one of the GREAT v...      1
1  One of the best game music soundtracks - for a...      1
2  Batteries died within a year ...: I bought thi...      0
3  works fine, but Maha Energy is better: Check o...      1
4  Great for the non-audiophile: Reviewed quite a...      1
```

In [4]:

```
print("Train Label Length: ", len(train_labels))
print("Train Text Length: ", len(train_texts))
print("Test Label Length: ", len(test_labels))
print("Test Text Length: ", len(test_texts))
```

```
Train Label Length:  36000
Train Text Length:  3600000
Test Label Length:  4000
Test Text Length:  400000
```

Data Cleaning

Removing Special Characters

In [5]:

```
def clean_text(text):
    # Remove non-alphanumeric characters and extra whitespace
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Convert multiple whitespace characters to a single space
    text = re.sub(r'\s+', ' ', text)
    # Convert the text to lowercase
    text = text.lower()
    return text

#train_df['text']=clean_text(train_df['text'])
#test_df['text']=clean_text(test_df['text'])
```

In [6]:

```
train_labels[0], train_texts[0]
```

Out[6]:

```
(1,
 'Stuning even for the non-gamer: This sound track was beautiful! It paints the senery in your mind so well I w
ould recomend it even to people who hate vid. game music! I have played the game Chrono Cross but out of all of
the games I have ever played it has the best music! It backs away from crude keyboarding and takes a fresher st
ep with grate guitars and soulful orchestras. It would impress anyone who cares to listen! ^_^')
```

In [7]:

```
train_labels[0], clean_text(train_texts[0])
```

Out[7]:

```
(1,
 'stuning even for the nongamer this sound track was beautiful it paints the senery in your mind so well i woul
d recomend it even to people who hate vid game music i have played the game chrono cross but out of all of the
games i have ever played it has the best music it backs away from crude keyboarding and takes a fresher step wi
th grate guitars and soulful orchestras it would impress anyone who cares to listen ')
```

In [8]:

```
test_labels[0], test_texts[0]
```

Out[8]:

```
(1,
 'Great CD: My lovely Pat has one of the GREAT voices of her generation. I have listened to this CD for YEARS a
nd I still LOVE IT. When I\'m in a good mood it makes me feel better. A bad mood just evaporates like sugar in
the rain. This CD just oozes LIFE. Vocals are jusat STUUNNING and lyrics just kill. One of life\'s hidden gems.
This is a desert isle CD in my book. Why she never made it big is just beyond me. Everytime I play this, no mat
ter black, white, young, old, male, female EVERYBODY says one thing "Who was that singing ?"')
```

In [9]:

```
test_labels[0], clean_text(test_texts[0])
```

Out[9]:

```
(1,
 'great cd my lovely pat has one of the great voices of her generation i have listened to this cd for years and
i still love it when im in a good mood it makes me feel better a bad mood just evaporates like sugar in the rai
n this cd just oozes life vocals are jusat stuunning and lyrics just kill one of lifes hidden gems this is a de
sert isle cd in my book why she never made it big is just beyond me everytime i play this no matter black white
young old male female everybody says one thing who was that singing ')
```

Train Label Count

In [10]:

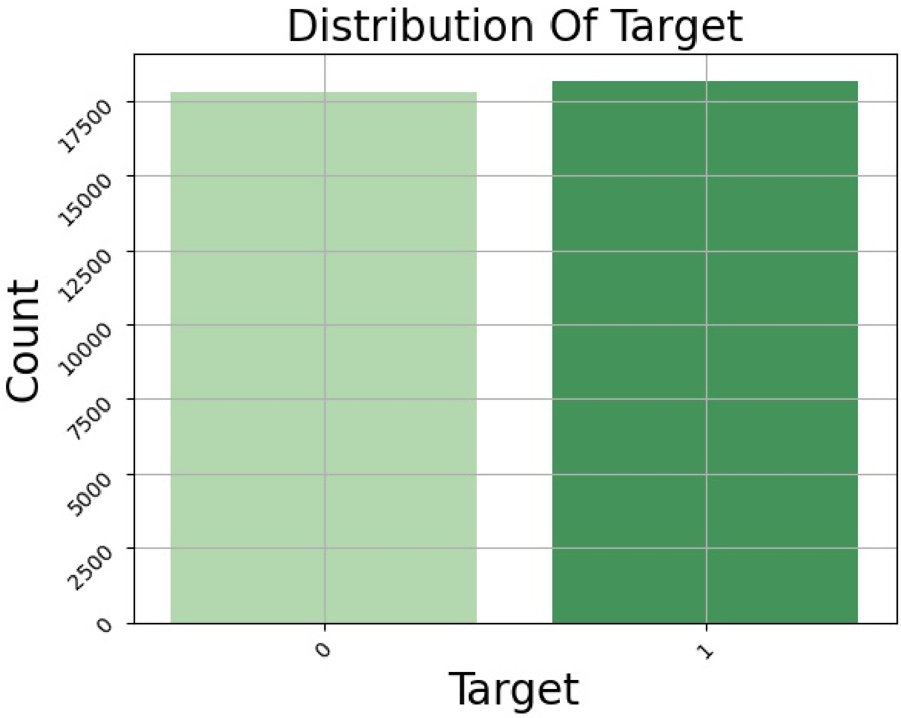
```
pd.DataFrame(train_labels).value_counts()
```

Out[10]:

```
1    18180
0    17820
dtype: int64
```

In [11]:

```
sns.countplot(x=pd.DataFrame(train_labels)[0],palette='Greens')
plt.title('Distribution Of Target',fontsize=20)
plt.xlabel('Target',fontsize=20)
plt.ylabel('Count',fontsize=20)
plt.grid(True)
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
```

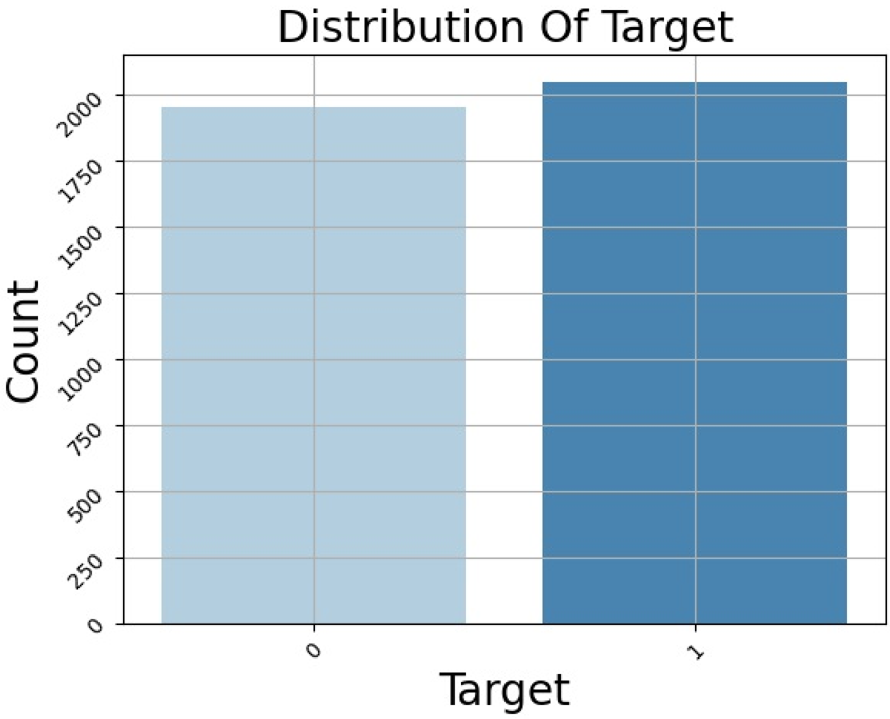


Test Label Count

```
In [12]: pd.DataFrame(test_labels).value_counts()
```

```
Out[12]: 1    2049
         0    1951
         dtype: int64
```

```
In [13]: sns.countplot(x=pd.DataFrame(test_labels)[0],palette='Blues')
         plt.title('Distribution Of Target',fontsize=20)
         plt.xlabel('Target',fontsize=20)
         plt.ylabel('Count',fontsize=20)
         plt.grid(True)
         plt.xticks(rotation=45)
         plt.yticks(rotation=45)
         plt.show()
```



```
In [14]: MAX_NB_WORDS = 10000
         MAX_SEQUENCE_LENGTH = 250
         EMBEDDING_DIM = 100
         tokenizer = text.Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~', lower=True)
         tokenizer.fit_on_texts(train_df['text'].values)
         word_index = tokenizer.word_index
         print('Found %s unique tokens.' % len(word_index))
```

```
Found 67675 unique tokens.
```

Tokenization

In [15]:

```
train_text = tokenizer.texts_to_sequences(train_df['text'].values)
train_text = pad_sequences(train_text, maxlen=MAX_SEQUENCE_LENGTH)
print('Data Tensor Shape:', train_text.shape)

y = pd.get_dummies(train_df['label']).values
print('Label Tensor Shape:', y.shape)
```

```
Data Tensor Shape: (36000, 250)
Label Tensor Shape: (36000, 2)
```

In [16]:

```
#max_words = 1000
#max_sequence_length = 100

#tokenizer = Tokenizer(num_words=max_words)
#tokenizer.fit_on_texts(train_texts)

#X_train = pad_sequences(X_train, maxlen=max_sequence_length)
#X_test = pad_sequences(X_test, maxlen=max_sequence_length)
```

Splitting the Data into Training and Testing Values

In [17]:

```
X_train, X_test, Y_train, Y_test = train_test_split(train_text, y, test_size=0.10, random_state=42)

print("X Training: ", X_train.shape)
print("Y Training: ", Y_train.shape)

print("X Testing: ", X_test.shape)
print("Y Testing: ", Y_test.shape)
```

```
X Training:  (32400, 250)
Y Training:  (32400, 2)
X Testing:  (3600, 250)
Y Testing:  (3600, 2)
```

Model Building

In [18]:

```
model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=train_text.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
# model.add(GlobalMaxPooling1D())
model.add(Dense(units=128, activation='relu'))
model.add(Dense(2, activation='softmax'))


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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 250, 100)          1000000   
                                                                 
 spatial_dropout1d (SpatialD  (None, 250, 100)         0         
 ropout1D)                                                       
                                                                 
 lstm (LSTM)                 (None, 100)               80400     
                                                                 
 dense (Dense)               (None, 128)               12928     
                                                                 
 dense_1 (Dense)             (None, 2)                 258       
                                                                 
=================================================================
Total params: 1,093,586
Trainable params: 1,093,586
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
epochs = 5

batch_size = 64

history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,validation_split=0.1,
                    callbacks=[EarlyStopping(monitor='val_loss', patience=3, min_delta=0.0001)])
```

```
Epoch 1/5
456/456 [==============================] - 467s 1s/step - loss: 0.3733 - accuracy: 0.8338 - val_loss: 0.2759 -
val_accuracy: 0.8880
Epoch 2/5
456/456 [==============================] - 449s 985ms/step - loss: 0.2475 - accuracy: 0.9046 - val_loss: 0.2890
- val_accuracy: 0.8827
Epoch 3/5
456/456 [==============================] - 433s 949ms/step - loss: 0.1783 - accuracy: 0.9346 - val_loss: 0.2908
- val_accuracy: 0.8938
Epoch 4/5
456/456 [==============================] - 418s 917ms/step - loss: 0.1376 - accuracy: 0.9507 - val_loss: 0.3020
- val_accuracy: 0.8938
```

In [20]:

```
y_pred = model.predict(X_test)
y_pred = (y_pred>0.5)
```

```
113/113 [==============================] - 4s 32ms/step
```

In [21]:

```
report = classification_report(Y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      1738
           1       0.89      0.90      0.90      1862

   micro avg       0.89      0.89      0.89      3600
   macro avg       0.89      0.89      0.89      3600
weighted avg       0.89      0.89      0.89      3600
 samples avg       0.89      0.89      0.89      3600
```

Let us test it on New Reviews (Both Positive and Negative Reviews)

In [22]:

```
text = "I really enjoyed this movie. The acting was great and the plot was engaging. Highly recommend!"
# preprocess the text data
# text = preprocess_text(text)
text_sequence = tokenizer.texts_to_sequences([text])
padded_sequence = pad_sequences(text_sequence, maxlen=MAX_SEQUENCE_LENGTH)
prediction = model.predict(padded_sequence)
predicted_class = np.argmax(prediction)
sentiment = "positive" if predicted_class == 1 else "negative"
print("The sentiment of the text is:", sentiment)
```

```
1/1 [==============================] - 0s 31ms/step
The sentiment of the text is: positive
```

In [23]:

```
text = "I was really disappointed with this hotel. The room was dirty and the staff was unhelpful."

text_sequence = tokenizer.texts_to_sequences([text])
padded_sequence = pad_sequences(text_sequence, maxlen=MAX_SEQUENCE_LENGTH)
prediction = model.predict(padded_sequence)
predicted_class = np.argmax(prediction)
sentiment = "positive" if predicted_class == 1 else "negative"
print("The sentiment of the text is:", sentiment)
```

```
1/1 [==============================] - 0s 16ms/step
The sentiment of the text is: negative
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js