NAME : ARUL KUMAR ARK

ROLL NO : 225229103

LAB - 1: Real time crawling of tweets from Twitter and creating a network of user mentions

Importing Dependencies

In [1]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np


import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer



from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder


import re

print("Tensorflow Version",tf.__version__)
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Tensorflow Version 2.1.0
```

Dataset Preprocessing

In [2]:
```
df = pd.read_csv('training.1600000.processed.noemoticon.csv',
                 encoding = 'latin',header=None)
df.head()
```

Out[2]:

	0	1	2	3	4	5
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

In [3]:
```
df.columns = ['sentiment', 'id', 'date', 'query', 'user_id', 'text']
df.head()
```

Out[3]:

	sentiment	id	date	query	user_id	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

In [4]:
```
df = df.drop(['id', 'date', 'query', 'user_id'], axis=1)
```

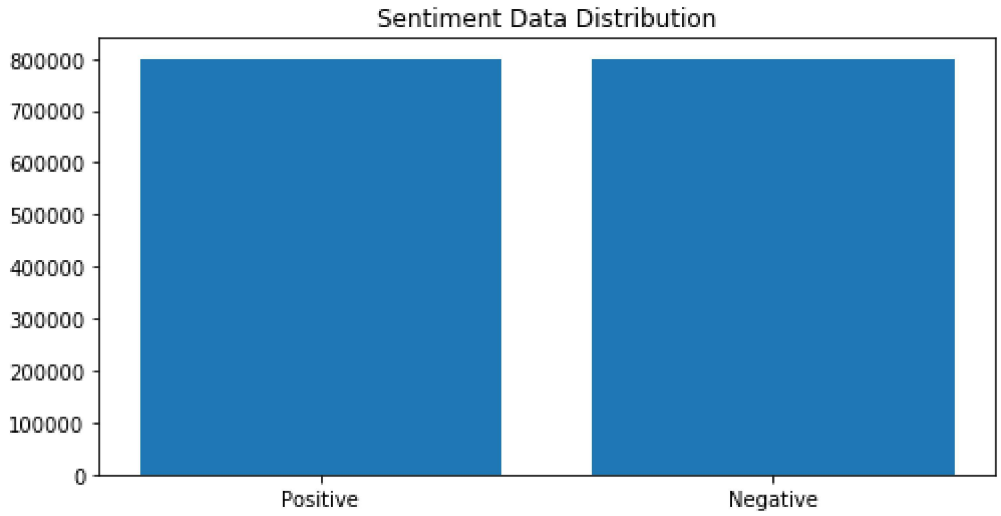
In [5]:
```
lab_to_sentiment = {0:"Negative", 4:"Positive"}
def label_decoder(label):
  return lab_to_sentiment[label]
df.sentiment = df.sentiment.apply(lambda x: label_decoder(x))
df.head()
```

Out[5]:

	sentiment	text
0	Negative	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	Negative	is upset that he can't update his Facebook by ...
2	Negative	@Kenichan I dived many times for the ball. Man...
3	Negative	my whole body feels itchy and like its on fire
4	Negative	@nationwideclass no, it's not behaving at all....

In [6]:
```
val_count = df.sentiment.value_counts()

plt.figure(figsize=(8,4))
plt.bar(val_count.index, val_count.values)
plt.title("Sentiment Data Distribution")
```

Out[6]:
```
Text(0.5, 1.0, 'Sentiment Data Distribution')
```



In [7]:
```
import random
random_idx_list = [random.randint(1,len(df.text)) for i in range(10)]
df.loc[random_idx_list,:].head(10)
```

Out[7]:

	sentiment	text
677110	Negative	@B0RNASTARtrell Oh no. IDK if I even wanna know.
1454647	Positive	@lilypenelope morning girl been reading... [j...
1508329	Positive	@BFeld13 I guess just the overall simplicity o...
1219837	Positive	@xscarletmx Pfft, they fail at being as awesom...
1554695	Positive	@riceagain Thanks for that link.. there's a lo...
1166888	Positive	shameless plugging http://rizzysanguinary.devi...
602781	Negative	I don't like being a big ole sneezy face.
699684	Negative	@yamerias Hope U Njoyed the movie - have 2 wai...
539358	Negative	my eyes feel so heavy .
448901	Negative	Just got burnt from my curling iron...

In [8]:
```
stop_words = stopwords.words('english')
stemmer = SnowballStemmer('english')

text_cleaning_re = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"
```

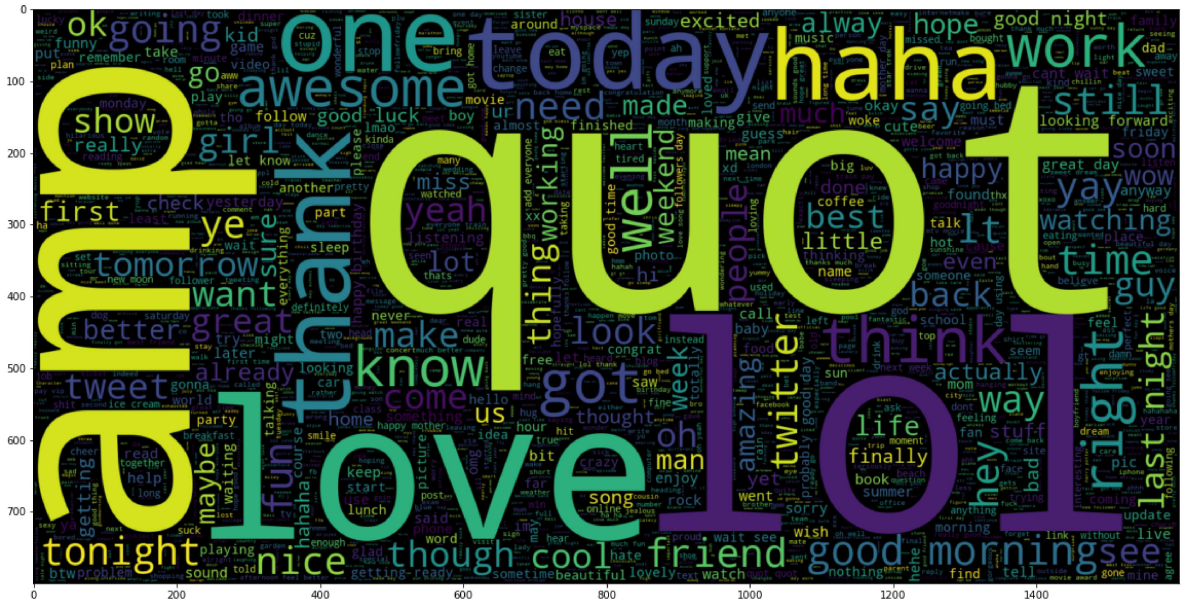
In [9]:
```
def preprocess(text, stem=False):
  text = re.sub(text_cleaning_re, ' ', str(text).lower()).strip()
  tokens = []
  for token in text.split():
    if token not in stop_words:
      if stem:
        tokens.append(stemmer.stem(token))
      else:
        tokens.append(token)
  return " ".join(tokens)
```

In [10]:
```
df.text = df.text.apply(lambda x: preprocess(x))
```

Positive Words

In [11]:
```
from wordcloud import WordCloud

plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".jc
plt.imshow(wc , interpolation = 'bilinear')
```

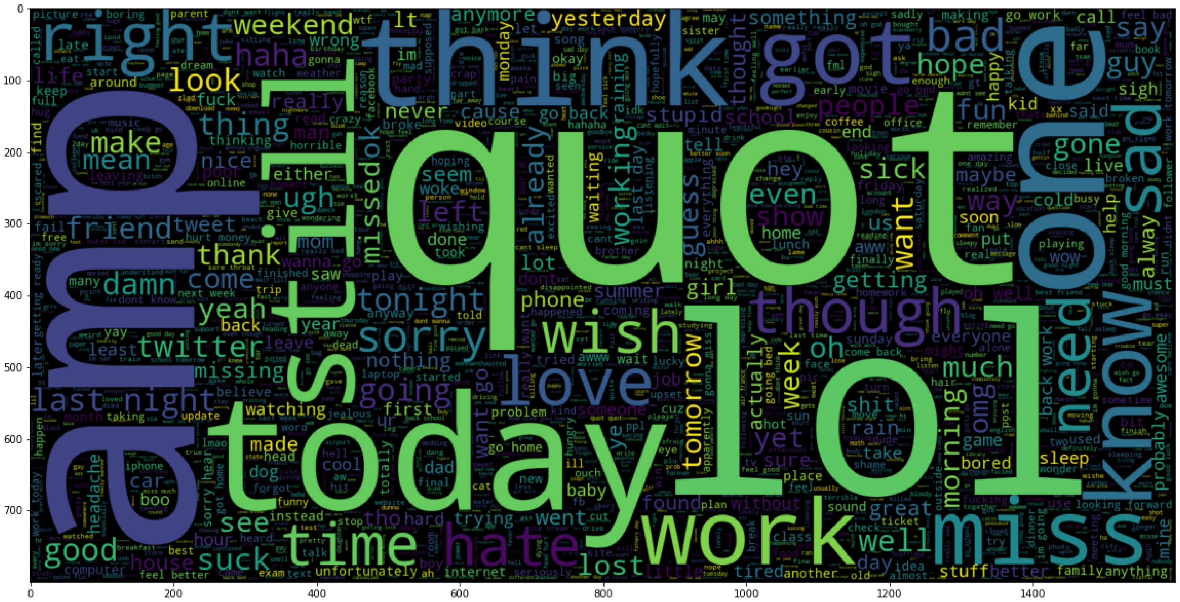
Out[11]:
```
<matplotlib.image.AxesImage at 0x7f377983f4d0>
```



Negative Words

In [12]:
```
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".jo
plt.imshow(wc , interpolation = 'bilinear')
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x7f37797ea690>
```



Train and Test Split

In [13]:
```
TRAIN_SIZE = 0.8
MAX_NB_WORDS = 100000
MAX_SEQUENCE_LENGTH = 30
```

In [14]:
```
train_data, test_data = train_test_split(df, test_size=1-TRAIN_SIZE,
                                         random_state=7)
print("Train Data size:", len(train_data))
print("Test Data size", len(test_data))
```

```
Train Data size: 1280000
Test Data size 320000
```

In [15]:
```
train_data.head(10)
```

Out[15]:

	sentiment	text
23786	Negative	need friends
182699	Negative	im trying call impossible
476661	Negative	good pace going 3k 13 min missed 5k turn ended...
1181490	Positive	u gonna shows ny soon luv see u live
878773	Positive	hell yea get em tattoos ink free wish parents ...
130866	Negative	yeah need 2 see ur mom calls back first rememb...
1235876	Positive	sounds like cup tea sign
717314	Negative	tired want sleep wtf
969880	Positive	amazing wish
748698	Negative	thank god wkrn abc affiliate nashville back mi...

Tokenization

In [16]:
```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_data.text)

word_index = tokenizer.word_index
vocab_size = len(tokenizer.word_index) + 1
print("Vocabulary Size :", vocab_size)
```

```
Using TensorFlow backend.

Vocabulary Size : 290575
```

In [17]:
```
from keras.preprocessing.sequence import pad_sequences

x_train = pad_sequences(tokenizer.texts_to_sequences(train_data.text),
                        maxlen = MAX_SEQUENCE_LENGTH)
x_test = pad_sequences(tokenizer.texts_to_sequences(test_data.text),
                       maxlen = MAX_SEQUENCE_LENGTH)

print("Training X Shape:",x_train.shape)
print("Testing X Shape:",x_test.shape)
```

```
Training X Shape: (1280000, 30)
Testing X Shape: (320000, 30)
```

In [18]:
```
labels = train_data.sentiment.unique().tolist()
```

Label Encoding

In [19]:
```
encoder = LabelEncoder()
encoder.fit(train_data.sentiment.to_list())

y_train = encoder.transform(train_data.sentiment.to_list())
y_test = encoder.transform(test_data.sentiment.to_list())

y_train = y_train.reshape(-1,1)
y_test = y_test.reshape(-1,1)

print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
y_train shape: (1280000, 1)
y_test shape: (320000, 1)
```

Word Emdedding

In [32]:
```
GLOVE_EMB = '/kaggle/working/glove.6B.300d.txt'
EMBEDDING_DIM = 300
LR = 1e-3
BATCH_SIZE = 1024
EPOCHS = 10
MODEL_PATH = '.../output/kaggle/working/best_model.hdf5'
```

In [24]:
```
embeddings_index = {}

f = open(GLOVE_EMB)
for line in f:
  values = line.split()
  word = value = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()

print('Found %s word vectors.' %len(embeddings_index))
```

```
Found 400000 word vectors.
```

In [25]:
```
embedding_matrix = np.zeros((vocab_size, EMBEDDING_DIM))
for word, i in word_index.items():
  embedding_vector = embeddings_index.get(word)
  if embedding_vector is not None:
    embedding_matrix[i] = embedding_vector
```

In [26]:
```
embedding_layer = tf.keras.layers.Embedding(vocab_size,
                                            EMBEDDING_DIM,
                                            weights=[embedding_matrix],
                                            input_length=MAX_SEQUENCE_LENGTH,
                                            trainable=False)
```

In [27]:
```
from tensorflow.keras.layers import Conv1D, Bidirectional, LSTM, Dense, Input,
from tensorflow.keras.layers import SpatialDropout1D
from tensorflow.keras.callbacks import ModelCheckpoint
```

In [28]:
```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = SpatialDropout1D(0.2)(embedding_sequences)
x = Conv1D(64, 5, activation='relu')(x)
x = Bidirectional(LSTM(64, dropout=0.2, recurrent_dropout=0.2))(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
outputs = Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(sequence_input, outputs)
```

In [34]:
```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau

model.compile(optimizer=Adam(learning_rate=LR), loss='binary_crossentropy',
              metrics=['accuracy'])
ReduceLROnPlateau = ReduceLROnPlateau(factor=0.1,
                                      min_lr = 0.01,
                                      monitor = 'val_loss',
                                      verbose = 1)
```

In [35]:
```
print("Training on GPU...") if tf.test.is_gpu_available() else print("Training
```

```
Training on GPU...
```

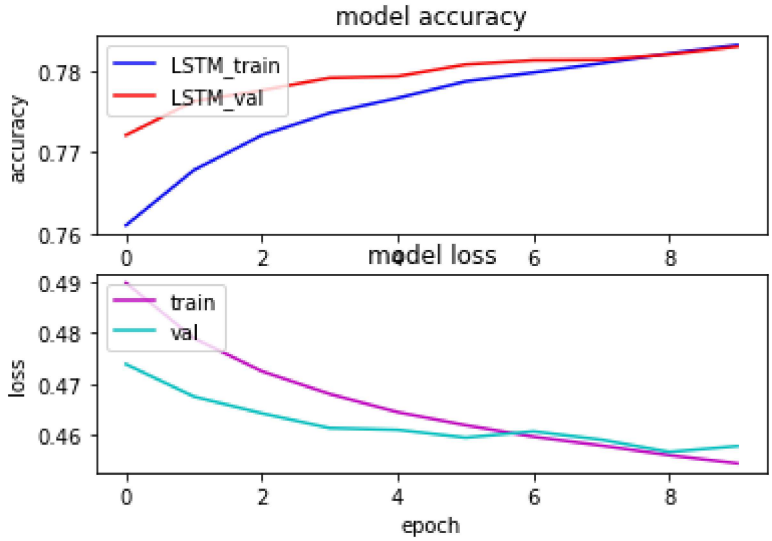
In [36]:
```
history = model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS,
                    validation_data=(x_test, y_test), callbacks=[ReduceLROnPla
```

```
Train on 1280000 samples, validate on 320000 samples
Epoch 1/10
1280000/1280000 [==============================] - 128s 100us/sample - loss:
0.4897 - accuracy: 0.7611 - val_loss: 0.4739 - val_accuracy: 0.7721
Epoch 2/10
1280000/1280000 [==============================] - 117s 92us/sample - loss:
0.4789 - accuracy: 0.7678 - val_loss: 0.4676 - val_accuracy: 0.7763
Epoch 3/10
1280000/1280000 [==============================] - 115s 90us/sample - loss:
0.4725 - accuracy: 0.7721 - val_loss: 0.4643 - val_accuracy: 0.7776
Epoch 4/10
1280000/1280000 [==============================] - 115s 90us/sample - loss:
0.4681 - accuracy: 0.7749 - val_loss: 0.4614 - val_accuracy: 0.7792
Epoch 5/10
1280000/1280000 [==============================] - 115s 90us/sample - loss:
0.4645 - accuracy: 0.7767 - val_loss: 0.4611 - val_accuracy: 0.7793
Epoch 6/10
1280000/1280000 [==============================] - 115s 90us/sample - loss:
0.4620 - accuracy: 0.7787 - val_loss: 0.4596 - val_accuracy: 0.7808
Epoch 7/10
1280000/1280000 [==============================] - 115s 90us/sample - loss:
0.4597 - accuracy: 0.7798 - val_loss: 0.4608 - val_accuracy: 0.7813
Epoch 8/10
1280000/1280000 [==============================] - 114s 89us/sample - loss:
0.4580 - accuracy: 0.7809 - val_loss: 0.4591 - val_accuracy: 0.7814
Epoch 9/10
1280000/1280000 [==============================] - 113s 89us/sample - loss:
0.4561 - accuracy: 0.7821 - val_loss: 0.4568 - val_accuracy: 0.7820
Epoch 10/10
1280000/1280000 [==============================] - 114s 89us/sample - loss:
0.4545 - accuracy: 0.7832 - val_loss: 0.4579 - val_accuracy: 0.7830
```

Model Evaluation

In [61]:
```
s, (at, al) = plt.subplots(2,1)
at.plot(history.history['accuracy'], c= 'b')
at.plot(history.history['val_accuracy'], c='r')
at.set_title('model accuracy')
at.set_ylabel('accuracy')
at.set_xlabel('epoch')
at.legend(['LSTM_train', 'LSTM_val'], loc='upper left')

al.plot(history.history['loss'], c='m')
al.plot(history.history['val_loss'], c='c')
al.set_title('model loss')
al.set_ylabel('loss')
al.set_xlabel('epoch')
al.legend(['train', 'val'], loc = 'upper left')
```

Out[61]: <matplotlib.legend.Legend at 0x7f3739c12490>



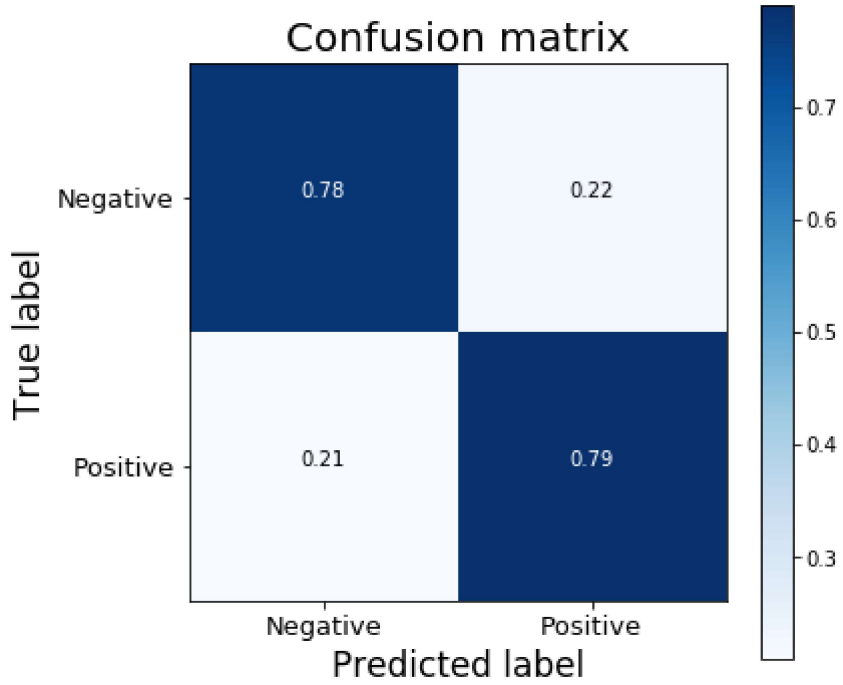
In [62]:
```
def decode_sentiment(score):
    return "Positive" if score>0.5 else "Negative"


scores = model.predict(x_test, verbose=1, batch_size=10000)
y_pred_1d = [decode_sentiment(score) for score in scores]
```

```
320000/320000 [==============================] - 4s 14us/sample
```

Confusion Matrix

In [63]:
```
import itertools
from sklearn.metrics import confusion_matrix, classification_report, accuracy_
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=20)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, fontsize=13)
    plt.yticks(tick_marks, classes, fontsize=13)

    fmt = '.2f'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label', fontsize=17)
    plt.xlabel('Predicted label', fontsize=17)
```

In [64]:
```
cnf_matrix = confusion_matrix(test_data.sentiment.to_list(), y_pred_1d)
plt.figure(figsize=(6,6))
plot_confusion_matrix(cnf_matrix, classes=test_data.sentiment.unique(), title=
plt.show()
```



Classification Scores

In [65]:
```
print(classification_report(list(test_data.sentiment), y_pred_1d))
```

```
              precision    recall  f1-score   support

    Negative       0.79      0.78      0.78    160542
    Positive       0.78      0.79      0.78    159458

    accuracy                           0.78    320000
   macro avg       0.78      0.78      0.78    320000
weighted avg       0.78      0.78      0.78    320000
```

In [ ]:
```

```