In [ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

In [ ]:
```
# DataFrame
import pandas as pd

# Matplot
import matplotlib.pyplot as plt
%matplotlib inline

# Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.manifold import TSNE
from sklearn.feature_extraction.text import TfidfVectorizer

# Keras
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout, Embedding, Flatten, Conv1D, MaxPool
from keras import utils
from keras.callbacks import ReduceLROnPlateau, EarlyStopping

# nltk
import nltk
from nltk.corpus import stopwords
from  nltk.stem import SnowballStemmer

# Word2vec
import gensim

# Utility
import re
import numpy as np
import os
from collections import Counter
import logging
import time
import pickle
import itertools

# Set log
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.IN
```

In [ ]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[ ]:
```
True
```

In [ ]:
```
DATASET_COLUMNS = ["target", "ids", "date", "flag", "user", "text"]
DATASET_ENCODING = "ISO-8859-1"
TRAIN_SIZE = 0.8


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

```
W2V_SIZE = 300
W2V_WINDOW = 7
W2V_EPOCH = 32
W2V_MIN_COUNT = 10


SEQUENCE_LENGTH = 300
EPOCHS = 8
BATCH_SIZE = 1024

POSITIVE = "POSITIVE"
NEGATIVE = "NEGATIVE"
NEUTRAL = "NEUTRAL"
SENTIMENT_THRESHOLDS = (0.4, 0.7)

KERAS_MODEL = "modeltest2_gru.h5"
WORD2VEC_MODEL = "modeltest2_gru.w2v"
TOKENIZER_MODEL = "tokenizertest2_gru.pkl"
ENCODER_MODEL = "encodertest2_gru.pkl"
```

In [ ]:
```
df = pd.read_csv('/content/drive/MyDrive/Dataset/training.1600000.processed.noemoticon.c
```

In [ ]:
```
df.head()
```

Out[ ]:

	target	ids	date	flag	user	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 [ ]:
```
df = df.sample(frac=1)
```

In [ ]:
```
df.head(50)
```

Out[ ]:

	target	ids	date	flag	user	text
1123088	4	1974444521	Sat May 30 13:08:59 PDT 2009	NO_QUERY	lweeks63	getting ready for work... Lisa
1432613	4	2060294404	Sat Jun 06 18:38:08 PDT 2009	NO_QUERY	magcampos	still hopefull for my Cubbies Have not missed...
1084650	4	1969029244	Fri May 29 22:50:24 PDT 2009	NO_QUERY	Jackdog24	@katmanduherself no lol cody is trying to sex...
1233585	4	1992201353	Mon Jun 01 08:22:08 PDT 2009	NO_QUERY	kenny277	One Night in Bangkok is THE BEST!!!! *DANCE*
176523	0	1965378850	Fri May 29 15:48:59 PDT 2009	NO_QUERY	Dogbook	Harley passed away in December 2008 http://ap...
361486	0	2046932144	Fri Jun 05 12:54:19 PDT 2009	NO_QUERY	daniellerangel	such a beautiful day outside and I am inside t...
1138260	4	1976827868	Sat May 30 19:12:12 PDT 2009	NO_QUERY	cjashton	is packed and ready to move
1528817	4	2177447407	Mon Jun 15 06:07:21 PDT 2009	NO_QUERY	carlos_teran	@Ali167 Have a wonderful Monday, Ali.
1213983	4	1989402130	Mon Jun 01 01:24:27 PDT 2009	NO_QUERY	MusicADdicts	@FROactive I think we did meet if I'm not mist...
950520	4	1824018819	Sat May 16 23:45:25 PDT 2009	NO_QUERY	poijakosalem	I just made Caesar Salad dressing. yumm.
1440304	4	2061649477	Sat Jun 06 21:18:48 PDT 2009	NO_QUERY	leimuise	Watching a docu on snow monkeys... our cousins
979528	4	1833939264	Mon May 18 01:00:56 PDT 2009	NO_QUERY	e_howson	It's (new, hot pink) ugg-boots, cup of tea and...
1125961	4	1974943568	Sat May 30 14:07:24 PDT 2009	NO_QUERY	AmanderFlander	Go go go! Check screening times
722817	0	2261609128	Sat Jun 20 21:03:30 PDT 2009	NO_QUERY	caromarie217	@ddlovato ME EITHER! not fair..
1348069	4	2044767310	Fri Jun 05 09:47:48 PDT 2009	NO_QUERY	thaibarcella	@Dannymcfly danny! please say happy b-day to m...
1087365	4	1969418184	Fri May 29 23:58:48 PDT 2009	NO_QUERY	chasecarnivine	Talking to the love of my life!! scott, i lo...
1241438	4	1994014556	Mon Jun 01 11:16:44 PDT 2009	NO_QUERY	AbbyLipstick	@BenJoBubble LOL soooo true! and it's all gloo...
1180714	4	1981967218	Sun May 31 10:15:22 PDT 2009	NO_QUERY	bootblackangela	Good afternoon everyone!!! Hope you all enjoy...
1382455	4	2052467087	Sat Jun 06 00:49:48 PDT 2009	NO_QUERY	MCeeYOSHi	@Julzeehope Well, i dont eat fast food.Low amo...

	target	ids	date	flag	user	text
60013	0	1686285031	Sun May 03 04:43:11 PDT 2009	NO_QUERY	eclaiire	Studying today Bad times.
198657	0	1971269543	Sat May 30 06:40:53 PDT 2009	NO_QUERY	KingCharlesI	@maleekberry yeh mate im gonna hound him next ...
630484	0	2232007620	Thu Jun 18 19:19:31 PDT 2009	NO_QUERY	laurenclark9	@skk123 SOOO damn bored haha how was your day?
1507413	4	2174490753	Sun Jun 14 22:39:50 PDT 2009	NO_QUERY	alexkoutek	fantastic day at the beach, the boardwalk and ...
236262	0	1979949992	Sun May 31 05:08:24 PDT 2009	NO_QUERY	retro_seventies	Back at the gulags again for another week. Ank...
752844	0	2286675267	Mon Jun 22 16:46:15 PDT 2009	NO_QUERY	yesimthatcool	I just cut my hair
891147	4	1689007899	Sun May 03 12:24:49 PDT 2009	NO_QUERY	N1ChrisBrownFan	@ditBOMB OMG! lol.....i don't have a clue why...
1352000	4	2046236714	Fri Jun 05 11:53:24 PDT 2009	NO_QUERY	JonitoB	Shall I buy a leopard gecko or a bearded drago...
1292060	4	2002986812	Tue Jun 02 05:37:56 PDT 2009	NO_QUERY	SapirAzoulay	In rehearsals . Rehearsals, rehearsals..... I ...
629454	0	2231648786	Thu Jun 18 18:50:28 PDT 2009	NO_QUERY	mhhughes	@MartyKFarris Where are you going? Everyone is...
127425	0	1834714502	Mon May 18 03:57:38 PDT 2009	NO_QUERY	lucy_r	@zeeblet yeah, sorry, only available to those ...
852282	4	1572936219	Mon Apr 20 23:05:17 PDT 2009	NO_QUERY	Miss_Ashleyyy	finished my paper. score! off to get some well...
951795	4	1824311027	Sun May 17 00:50:56 PDT 2009	NO_QUERY	katzirra	Weird to say but Calvin's apartment? He hangs ...
1044458	4	1957445461	Fri May 29 00:31:27 PDT 2009	NO_QUERY	Kaattt11	@CheMerf Heard your greeting for archiejoepet.
107087	0	1823808962	Sat May 16 23:03:13 PDT 2009	NO_QUERY	cpf	@SquireFred We saw you and @monkiyo on stage a...
724959	0	2262205247	Sat Jun 20 22:04:27 PDT 2009	NO_QUERY	gabbycastillo	@JoelMadden I am really jealous you got to see...
304889	0	1999854124	Mon Jun 01 20:56:47 PDT 2009	NO_QUERY	gfisher24	help me...my foot is throbbin after i got the ...
953405	4	1824639615	Sun May 17 02:16:01 PDT 2009	NO_QUERY	bridaisy	Just left tji fridays ! And i think we found a...
163535	0	1958068448	Fri May 29 02:33:06 PDT 2009	NO_QUERY	allendoggie	tomorrow will be my last day in work.. how sad...

	target	ids	date	flag	user	text
184692	0	1967774772	Fri May 29 20:11:19 PDT 2009	NO_QUERY	ChipperDuck	At a wedding... No hot guys...
1458590	4	2063679018	Sun Jun 07 03:38:43 PDT 2009	NO_QUERY	StevieDavidson	I love new music
513737	0	2190458590	Tue Jun 16 02:47:19 PDT 2009	NO_QUERY	Ambcompte	@MRGOULD Why are you not coming to Spain?
1447293	4	2062604915	Sat Jun 06 23:43:41 PDT 2009	NO_QUERY	DeniseVlogs	@Joppy I'm cool with that... and don't forget ...
1033128	4	1936081443	Wed May 27 07:25:18 PDT 2009	NO_QUERY	beijingfairmont	We DIGG this "Digg Adds Twitter and Face...
386457	0	2053797399	Sat Jun 06 05:43:51 PDT 2009	NO_QUERY	hollyalyxfinch	@ScruffyPanther You so are. Poor B
830256	4	1557283719	Sun Apr 19 03:08:23 PDT 2009	NO_QUERY	Nhipham	Is in bed on her touch and can hear loud as mu...
69660	0	1693375211	Sun May 03 22:21:26 PDT 2009	NO_QUERY	eerational	wants/needs to take another nap
899881	4	1693934386	Mon May 04 00:21:52 PDT 2009	NO_QUERY	zulunationn	so, its no longer the day of my birth. and i k...
1511310	4	2175078873	Mon Jun 15 00:01:59 PDT 2009	NO_QUERY	SeattleEconomy	@sherpaco no matter what u hear people say, Ko...
927240	4	1759124879	Sun May 10 18:31:02 PDT 2009	NO_QUERY	anyabast	@Gennita I'm pretty sure he wants to do that o...
917248	4	1753451907	Sun May 10 01:42:23 PDT 2009	NO_QUERY	feistyfrogg	can't believe the weekend is over already! Wh...

In [ ]:
```
print("Dataset size:", len(df))
```

```
Dataset size: 1600000
```

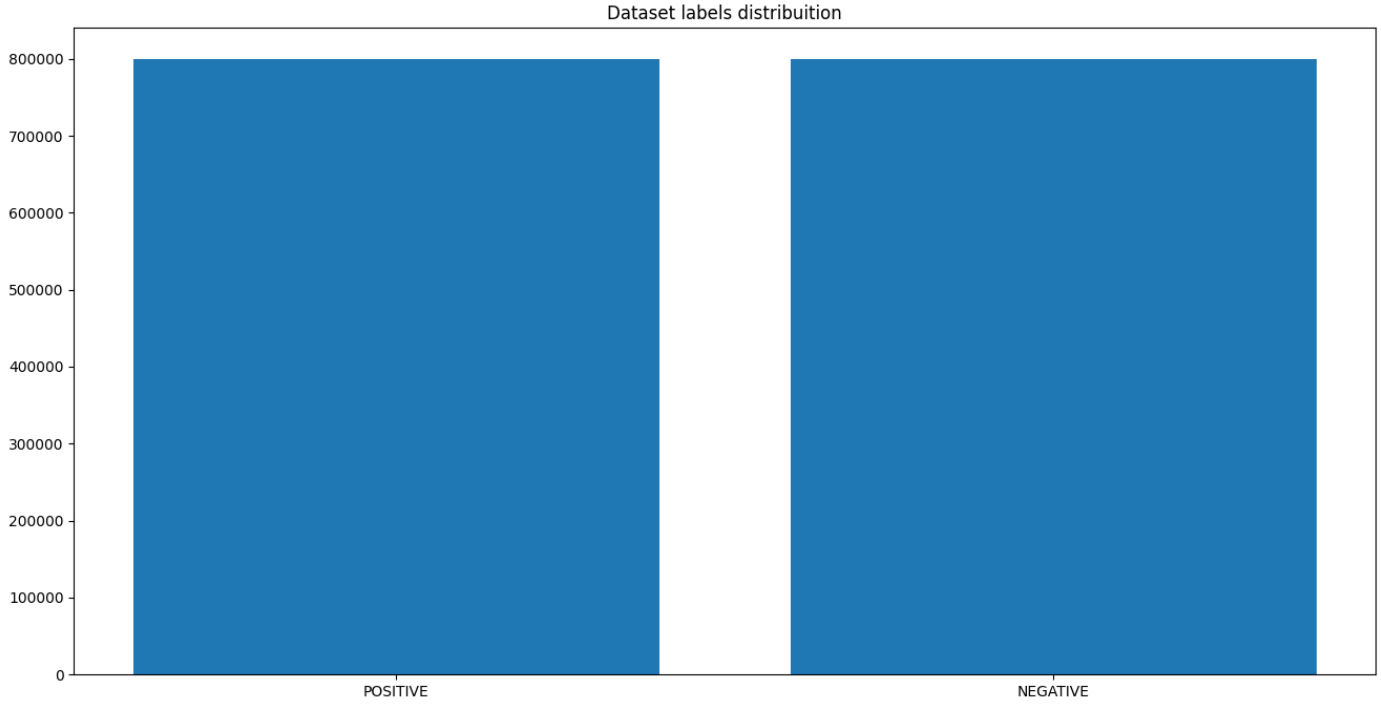
In [ ]:
```
decode_map = {0: "NEGATIVE", 2: "NEUTRAL", 4: "POSITIVE"}
def decode_sentiment(label):
    return decode_map[int(label)]
```

In [ ]:
```
%%time
df.target = df.target.apply(lambda x: decode_sentiment(x))
```

```
CPU times: user 815 ms, sys: 17.5 ms, total: 833 ms
Wall time: 874 ms
```

In [ ]:
```
target_cnt = Counter(df.target)

plt.figure(figsize=(16,8))
plt.bar(target_cnt.keys(), target_cnt.values())
plt.title("Dataset labels distribuition")
```

```
Text(0.5, 1.0, 'Dataset labels distribuition')
```



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

In [ ]:
```
def preprocess(text, stem=False):
    # Remove link,user and special characters
    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 [ ]:
```
%%time
df.text = df.text.apply(lambda x: preprocess(x))
```

```
CPU times: user 1min 8s, sys: 338 ms, total: 1min 8s
Wall time: 1min 16s
```

In [ ]:
```
df_train, df_test = train_test_split(df, test_size=1-TRAIN_SIZE, random_state=42)
print("TRAIN size:", len(df_train))
print("TEST size:", len(df_test))
```

```
TRAIN size: 1280000
TEST size: 320000
```

In [ ]:
```
%%time
documents = [_text.split() for _text in df_train.text]
```

```
CPU times: user 5.41 s, sys: 616 ms, total: 6.02 s
Wall time: 6.15 s
```

In [ ]:
```
w2v_model = gensim.models.word2vec.Word2Vec(vector_size=W2V_SIZE,
                                            window=W2V_WINDOW,
                                            min_count=W2V_MIN_COUNT,
                                            workers=8)
```

In [ ]:
```
w2v_model.build_vocab(documents)
```

In [ ]:
```
%%time
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df_train.text)

vocab_size = len(tokenizer.word_index) + 1
print("Total words", vocab_size)
```

```
Total words 290711
CPU times: user 30.1 s, sys: 250 ms, total: 30.3 s
Wall time: 48.6 s
```

In [ ]:
```
import io
import json

# Saving
tokenizer_json = tokenizer.to_json()
with io.open('tokenizer.json', 'w', encoding='utf-8') as f:
    f.write(json.dumps(tokenizer_json, ensure_ascii=False))
```

In [ ]:
```
%%time
x_train = pad_sequences(tokenizer.texts_to_sequences(df_train.text), maxlen=SEQUENCE_LEN
x_test = pad_sequences(tokenizer.texts_to_sequences(df_test.text), maxlen=SEQUENCE_LENGT
```

```
CPU times: user 29.8 s, sys: 786 ms, total: 30.6 s
Wall time: 32.3 s
```

In [ ]:
```
labels = df_train.target.unique().tolist()
labels.append(NEUTRAL)
labels
```

Out[ ]:
```
['NEGATIVE', 'POSITIVE', 'NEUTRAL']
```

In [ ]:
```
encoder = LabelEncoder()
encoder.fit(df_train.target.tolist())

y_train = encoder.transform(df_train.target.tolist())
y_test = encoder.transform(df_test.target.tolist())

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

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

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

In [ ]:
```
print("x_train", x_train.shape)
print("y_train", y_train.shape)
print()
print("x_test", x_test.shape)
print("y_test", y_test.shape)
```

```
x_train (1280000, 300)
y_train (1280000, 1)

x_test (320000, 300)
y_test (320000, 1)
```

In [ ]:
```
y_train[:10]
```

Out[ ]:
```
array([[0],
       [0],
       [1],
       [1],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0]])
```

In [ ]:
```
embedding_matrix = np.zeros((vocab_size, W2V_SIZE))
for word, i in tokenizer.word_index.items():
  if word in w2v_model.wv:
    embedding_matrix[i] = w2v_model.wv[word]
print(embedding_matrix.shape)
```

```
(290711, 300)
```

In [ ]:
```
embedding_layer = Embedding(vocab_size, W2V_SIZE, weights=[embedding_matrix], input_leng
```

Model For Trainning

Simple RNN

In [ ]:
```
from keras.layers import SimpleRNN

model = Sequential()
model.add(Embedding(vocab_size, 2, input_length=SEQUENCE_LENGTH))
model.add(SimpleRNN(32,return_sequences=False))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 300, 2)            581232

 simple_rnn (SimpleRNN)      (None, 32)                1120

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 582,385
Trainable params: 582,385
Non-trainable params: 0
_________________________________________________________________
```

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

In [ ]:
```
history = model.fit(x_train, y_train, batch_size=128, epochs=6, verbose=1, validation_sp
```

```
Epoch 1/6
8000/8000 [==============================] - 2055s 256ms/step - loss: 0.4925 - accuracy:
0.7607 - val_loss: 0.4692 - val_accuracy: 0.7792
Epoch 2/6
8000/8000 [==============================] - 2068s 258ms/step - loss: 0.4581 - accuracy:
0.7865 - val_loss: 0.4890 - val_accuracy: 0.7648
Epoch 3/6
8000/8000 [==============================] - 2081s 260ms/step - loss: 0.4464 - accuracy:
0.7944 - val_loss: 0.4790 - val_accuracy: 0.7746
Epoch 4/6
8000/8000 [==============================] - 2085s 261ms/step - loss: 0.4245 - accuracy:
0.8079 - val_loss: 0.4793 - val_accuracy: 0.7751
Epoch 5/6
8000/8000 [==============================] - 2118s 265ms/step - loss: 0.4095 - accuracy:
0.8162 - val_loss: 0.4872 - val_accuracy: 0.7668
Epoch 6/6
8000/8000 [==============================] - 2088s 261ms/step - loss: 0.4005 - accuracy:
0.8202 - val_loss: 0.5069 - val_accuracy: 0.7671
```

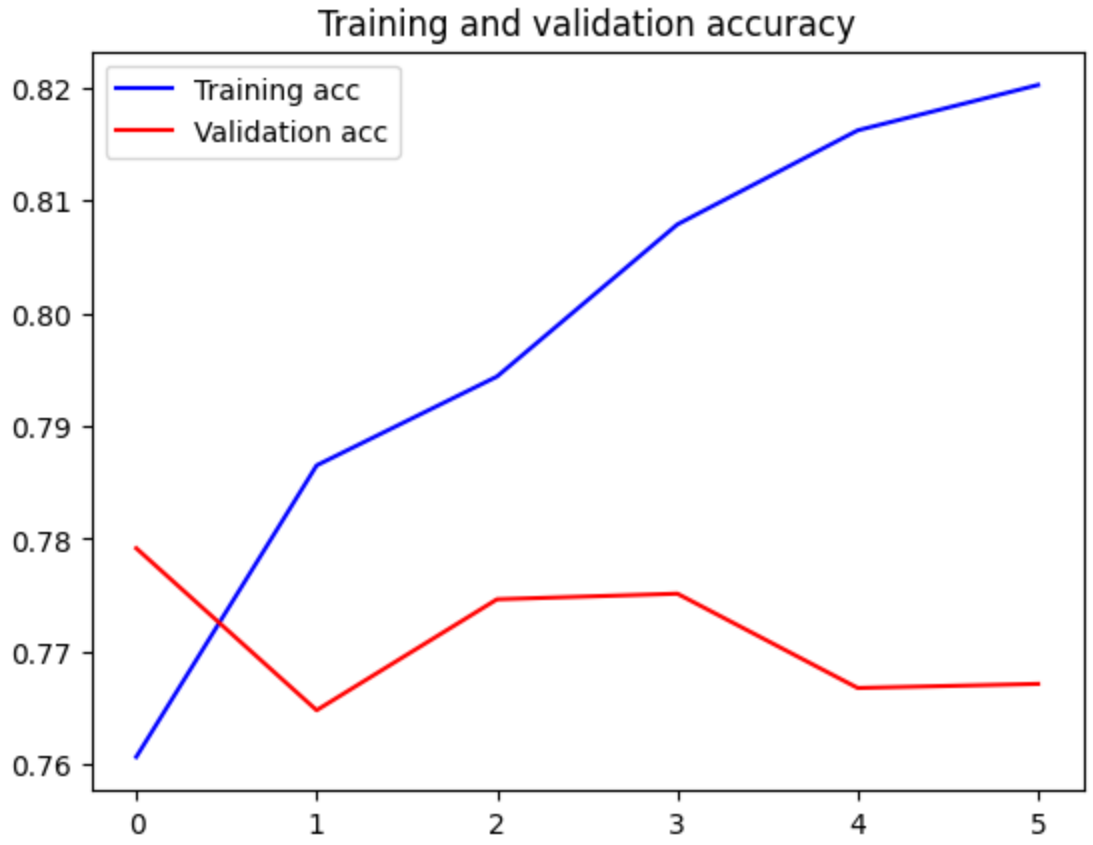
In [ ]:
```
%%time
score = model.evaluate(x_test, y_test, batch_size=BATCH_SIZE)
print()
print("ACCURACY:",score[1])
print("LOSS:",score[0])
```

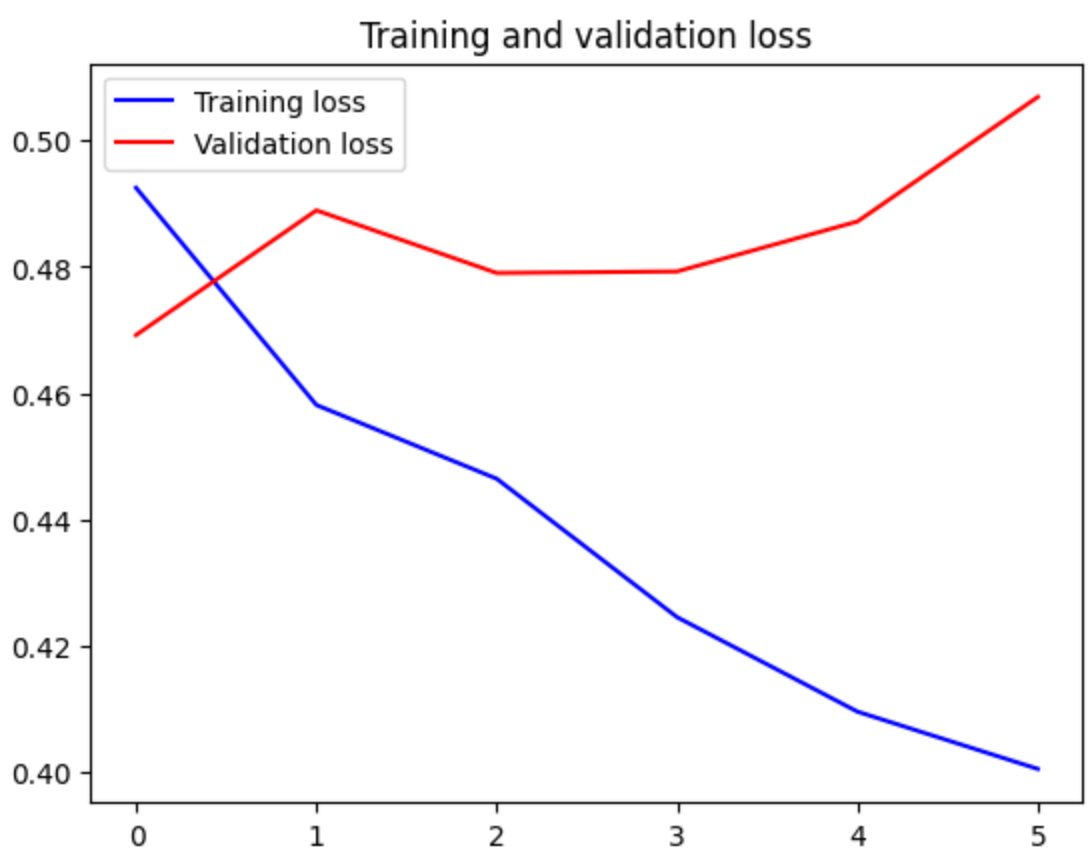
```
313/313 [==============================] - 9s 30ms/step - loss: 0.5073 - accuracy: 0.768
2

ACCURACY: 0.7681937217712402
LOSS: 0.5072503685951233
CPU times: user 9.44 s, sys: 301 ms, total: 9.74 s
Wall time: 10.7 s
```

In [ ]:
```
def plot_acc_loss(history):

    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)

    plt.plot(epochs, acc, 'bo', label = 'Training Accuracy')
    plt.plot(epochs, val_acc, 'r', label = 'Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.figure()
    plt.plot(epochs, loss, 'bo', label = 'Training Loss')
    plt.plot(epochs, val_loss, 'r', label = 'Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

In [ ]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))
```

```
plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [ ]:
```
def decode_sentiment(score, include_neutral=True):
    if include_neutral:
        label = NEUTRAL
        if score <= SENTIMENT_THRESHOLDS[0]:
            label = NEGATIVE
        elif score >= SENTIMENT_THRESHOLDS[1]:
            label = POSITIVE

        return label
    else:
        return NEGATIVE if score < 0.5 else POSITIVE
```

In [ ]:
```
def predict(text, include_neutral=True):
    start_at = time.time()
    # Tokenize text
    x_test = pad_sequences(tokenizer.texts_to_sequences([text]), maxlen=SEQUENCE_LENGTH)
    # Predict
    score = model.predict([x_test])[0]
    # Decode sentiment
    label = decode_sentiment(score, include_neutral=include_neutral)

    return {"label": label, "score": float(score),
       "elapsed_time": time.time()-start_at}
```

In [ ]:
```
predict("Shit happens today")
```

```
1/1 [==============================] - 0s 171ms/step
```

Out[ ]:
```
{'label': 'NEGATIVE',
 'score': 0.3375582695007324,
 'elapsed_time': 0.21554136276245117}
```

In [ ]:
```
predict("Today was my marriage aniversery and I proposed her again")
```

```
1/1 [==============================] - 0s 139ms/step
```

Out[ ]:
```
{'label': 'POSITIVE',
 'score': 0.9870448708534241,
 'elapsed_time': 0.21779179573059082}
```

In [ ]:
```
predict("I was kicked by the hotel manager because I made other guest unhappy")
```

Out[ ]:
```
1/1 [==============================] - 0s 43ms/step
{'label': 'NEGATIVE',
 'score': 0.04854927957057953,
 'elapsed_time': 0.07855057716369629}
```

In [ ]:
```
predict("She got pregnent because it was my fault . I dont want to be a father at this e
```

Out[ ]:
```
1/1 [==============================] - 0s 39ms/step
{'label': 'NEGATIVE',
 'score': 0.10528918355703354,
 'elapsed_time': 0.08156108856201172}
```

In [ ]:
```
predict("Got luckey to get a family like mine to be supportive arround me always")
```

Out[ ]:
```
1/1 [==============================] - 0s 121ms/step
{'label': 'POSITIVE',
 'score': 0.8843143582344055,
 'elapsed_time': 0.2693033218383789}
```

In [ ]:
```
predict("She said YES ")
```

Out[ ]:
```
1/1 [==============================] - 0s 61ms/step
{'label': 'POSITIVE',
 'score': 0.7906350493431091,
 'elapsed_time': 0.10763287544250488}
```

In [ ]:
```
predict("PUBG is now banned in India")
```

Out[ ]:
```
1/1 [==============================] - 0s 131ms/step
{'label': 'NEGATIVE',
 'score': 0.16254423558712006,
 'elapsed_time': 0.27535176277160645}
```

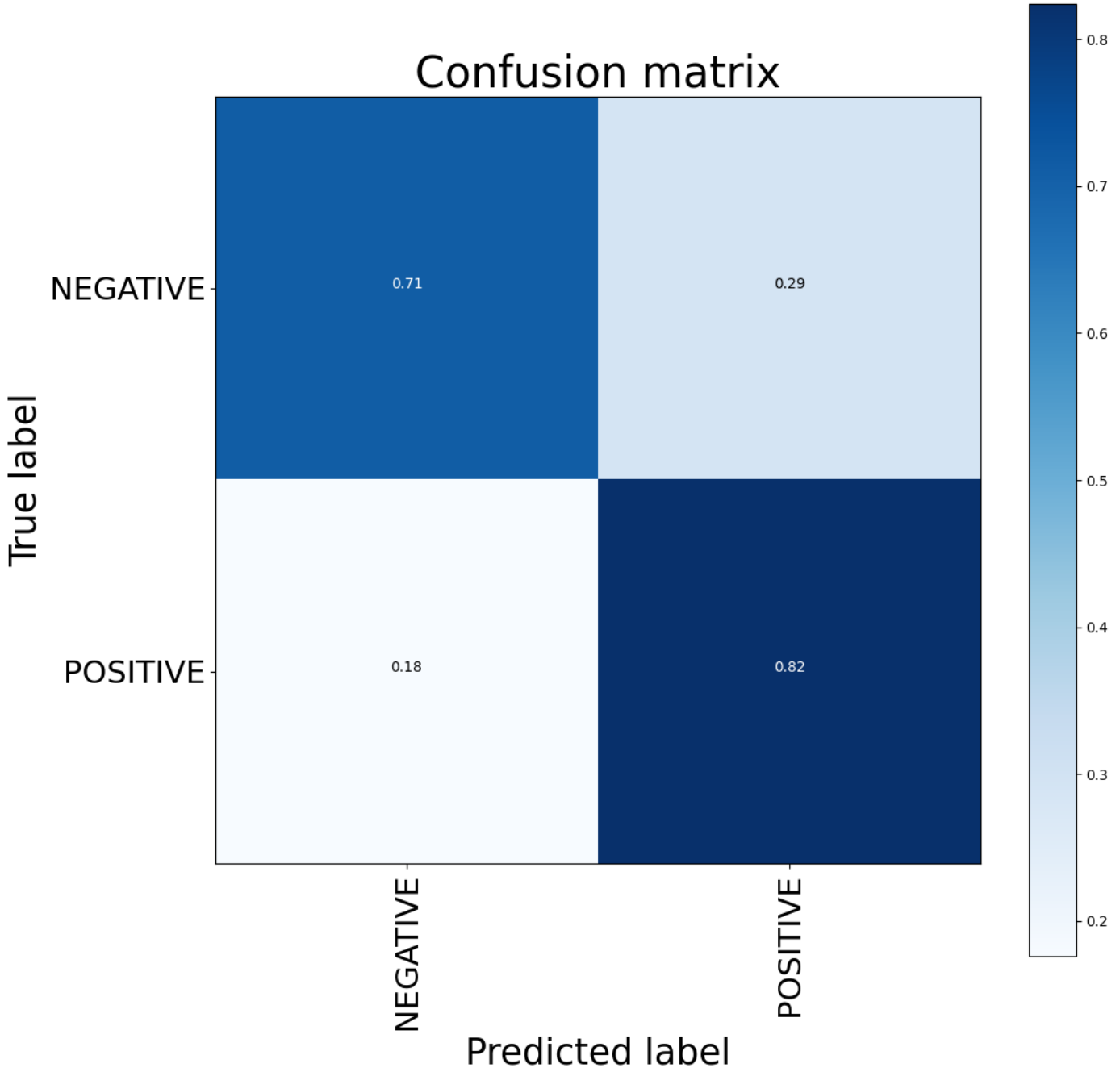
In [ ]:
```
%%time
y_pred_1d = []
y_test_1d = list(df_test.target)
scores = model.predict(x_test, verbose=1, batch_size=8000)
y_pred_1d = [decode_sentiment(score, include_neutral=False) for score in scores]
```

```
40/40 [==============================] - 1s 29ms/step
CPU times: user 2.09 s, sys: 330 ms, total: 2.42 s
Wall time: 2.32 s
```

In [ ]:
```
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=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90, fontsize=22)
    plt.yticks(tick_marks, classes, fontsize=22)
```

```
    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=25)
    plt.xlabel('Predicted label', fontsize=25)
```

In [ ]:
```
%%time

cnf_matrix = confusion_matrix(y_test_1d, y_pred_1d)
plt.figure(figsize=(12,12))
plot_confusion_matrix(cnf_matrix, classes=df_train.target.unique(), title="Confusion mat
plt.show()
```




```
CPU times: user 1.99 s, sys: 289 ms, total: 2.28 s
Wall time: 2.06 s
```

In [ ]:
```
print(classification_report(y_test_1d, y_pred_1d))
```

```
              precision    recall  f1-score   support

    NEGATIVE       0.80      0.71      0.75    159994
    POSITIVE       0.74      0.82      0.78    160006

    accuracy                           0.77    320000
   macro avg       0.77      0.77      0.77    320000
weighted avg       0.77      0.77      0.77    320000
```

In [ ]:
```
accuracy_score(y_test_1d, y_pred_1d)
```

Out[ ]:
```
0.76819375
```

In [ ]:
```
model.save(KERAS_MODEL)
w2v_model.save(WORD2VEC_MODEL)
pickle.dump(tokenizer, open(TOKENIZER_MODEL, "wb"), protocol=0)
pickle.dump(encoder, open(ENCODER_MODEL, "wb"), protocol=0)
```

GRU Model

In [ ]:
```
from keras.layers import GRU  # Import the GRU layer

model_gru = Sequential()
model_gru.add(Embedding(vocab_size, 2, input_length=SEQUENCE_LENGTH))
model_gru.add(GRU(32, return_sequences=False))  # Use GRU instead of SimpleRNN
model_gru.add(Dense(1, activation='sigmoid'))

model_gru.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 300, 2)            581594

 gru (GRU)                   (None, 32)                3456

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 585083 (2.23 MB)
Trainable params: 585083 (2.23 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
model_gru.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

In [ ]:
```
history = model_gru.fit(x_train, y_train, batch_size=128, epochs=6, verbose=1, validatio
```

```
Epoch 1/6
8000/8000 [==============================] - 155s 18ms/step - loss: 0.4814 - accuracy:
0.7687 - val_loss: 0.4599 - val_accuracy: 0.7821
Epoch 2/6
8000/8000 [==============================] - 103s 13ms/step - loss: 0.4405 - accuracy:
0.7946 - val_loss: 0.4576 - val_accuracy: 0.7835
Epoch 3/6
8000/8000 [==============================] - 98s 12ms/step - loss: 0.4227 - accuracy: 0.
8051 - val_loss: 0.4615 - val_accuracy: 0.7811
Epoch 4/6
8000/8000 [==============================] - 97s 12ms/step - loss: 0.4106 - accuracy: 0.
8114 - val_loss: 0.4651 - val_accuracy: 0.7802
Epoch 5/6
8000/8000 [==============================] - 97s 12ms/step - loss: 0.3972 - accuracy: 0.
8187 - val_loss: 0.4723 - val_accuracy: 0.7771
Epoch 6/6
8000/8000 [==============================] - 96s 12ms/step - loss: 0.3874 - accuracy: 0.
8239 - val_loss: 0.4814 - val_accuracy: 0.7747
```

In [ ]:

```
%%time
score = model_gru.evaluate(x_test, y_test, batch_size=BATCH_SIZE)
print()
print("ACCURACY:",score[1])
print("LOSS:",score[0])
```

```
313/313 [==============================] - 4s 8ms/step - loss: 0.4833 - accuracy: 0.7741

ACCURACY: 0.7741249799728394
LOSS: 0.48326563835144043
CPU times: user 3.1 s, sys: 588 ms, total: 3.69 s
Wall time: 5.28 s
```

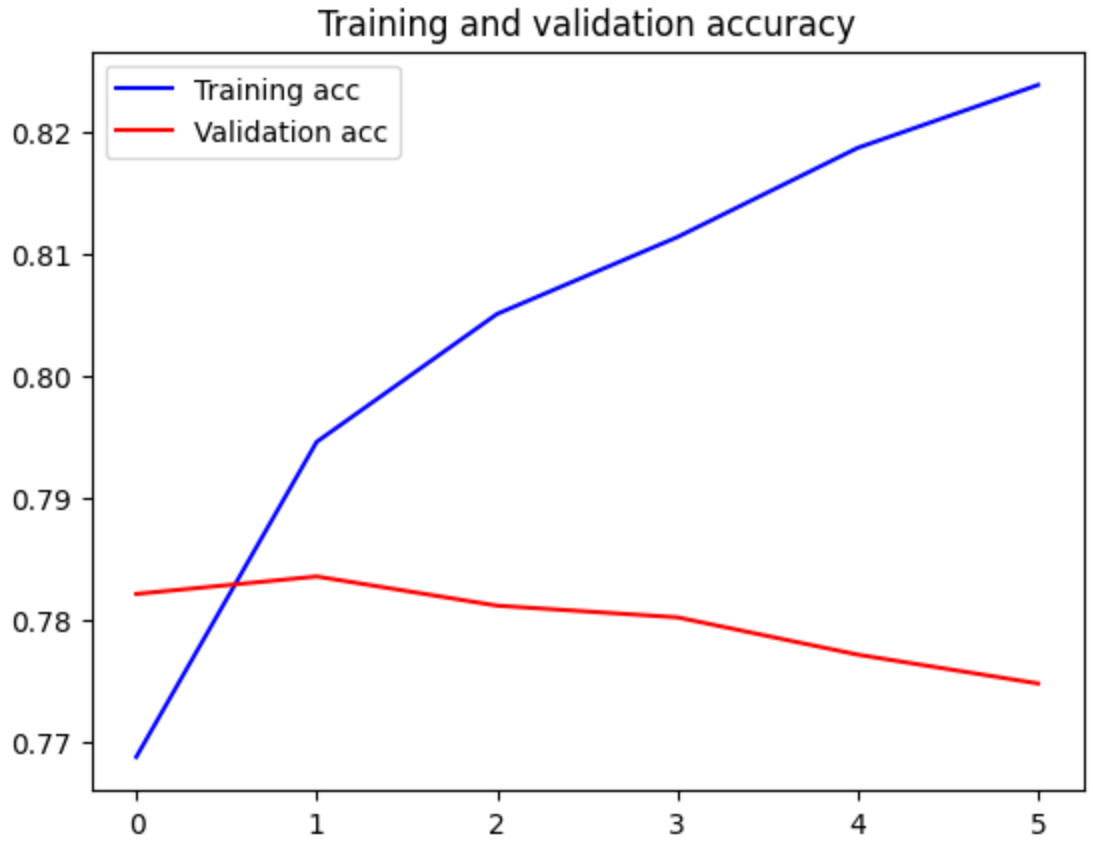
In [ ]:

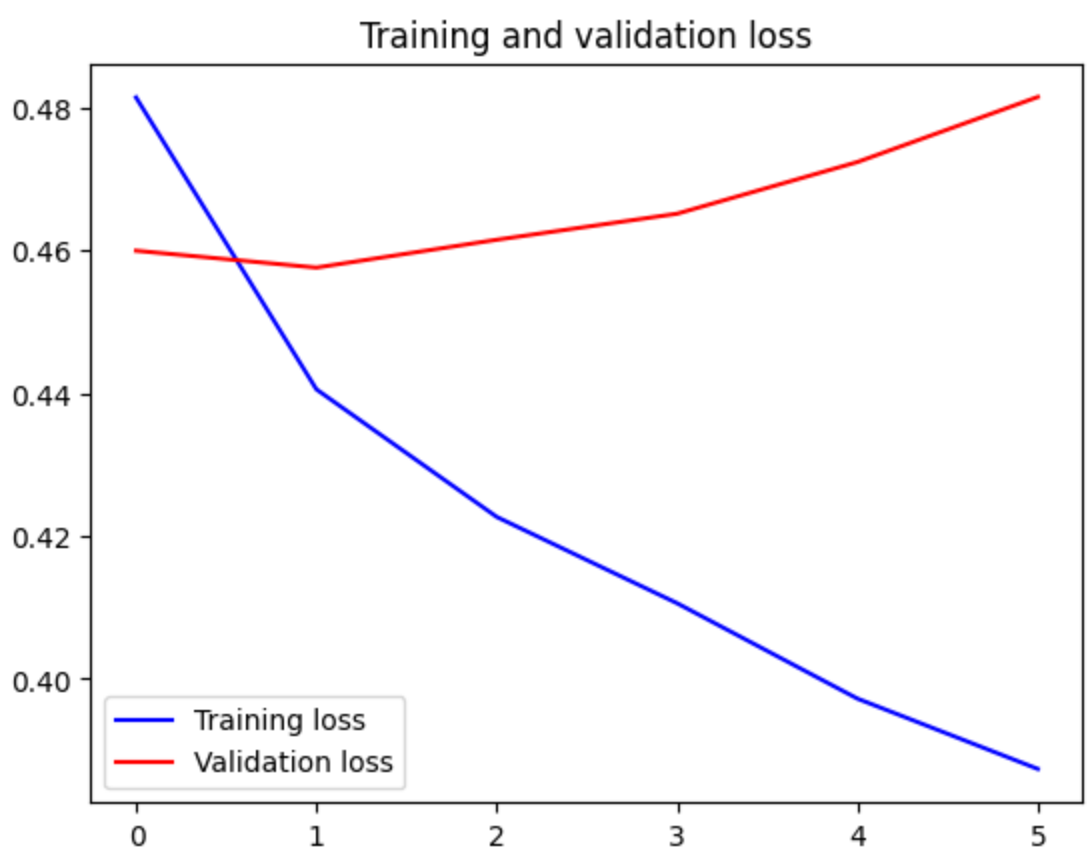
```
def plot_acc_loss(history):

    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)

    plt.plot(epochs, acc, 'bo', label = 'Training Accuracy')
    plt.plot(epochs, val_acc, 'r', label = 'Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.figure()
    plt.plot(epochs, loss, 'bo', label = 'Training Loss')
    plt.plot(epochs, val_loss, 'r', label = 'Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

In [ ]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training acc')
```

```
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [ ]:
```
def decode_sentiment(score, include_neutral=True):
    if include_neutral:
        label = NEUTRAL
        if score <= SENTIMENT_THRESHOLDS[0]:
            label = NEGATIVE
        elif score >= SENTIMENT_THRESHOLDS[1]:
            label = POSITIVE

        return label
    else:
        return NEGATIVE if score < 0.5 else POSITIVE
```

In [ ]:
```
def predict(text, include_neutral=True):
    start_at = time.time()
    # Tokenize text
    x_test = pad_sequences(tokenizer.texts_to_sequences([text]), maxlen=SEQUENCE_LENGTH)
    # Predict
    score = model_gru.predict([x_test])[0]
    # Decode sentiment
    label = decode_sentiment(score, include_neutral=include_neutral)

    return {"label": label, "score": float(score),
       "elapsed_time": time.time()-start_at}
```

In [ ]:
```
predict("I dont like to talk to anyone today.")
```

```
1/1 [==============================] - 0s 316ms/step
```

Out[ ]:
```
{'label': 'NEGATIVE',
 'score': 0.23090879619121552,
 'elapsed_time': 0.36948180198669434}
```

In [ ]:
```
predict("I am first in the race")
```

```
1/1 [==============================] - 0s 33ms/step
```

Out[ ]:
```
{'label': 'POSITIVE',
 'score': 0.8875918984413147,
 'elapsed_time': 0.09151721000671387}
```

In [ ]:
```
predict("My stomach is upset today")
```

```
1/1 [==============================] - 0s 33ms/step
```

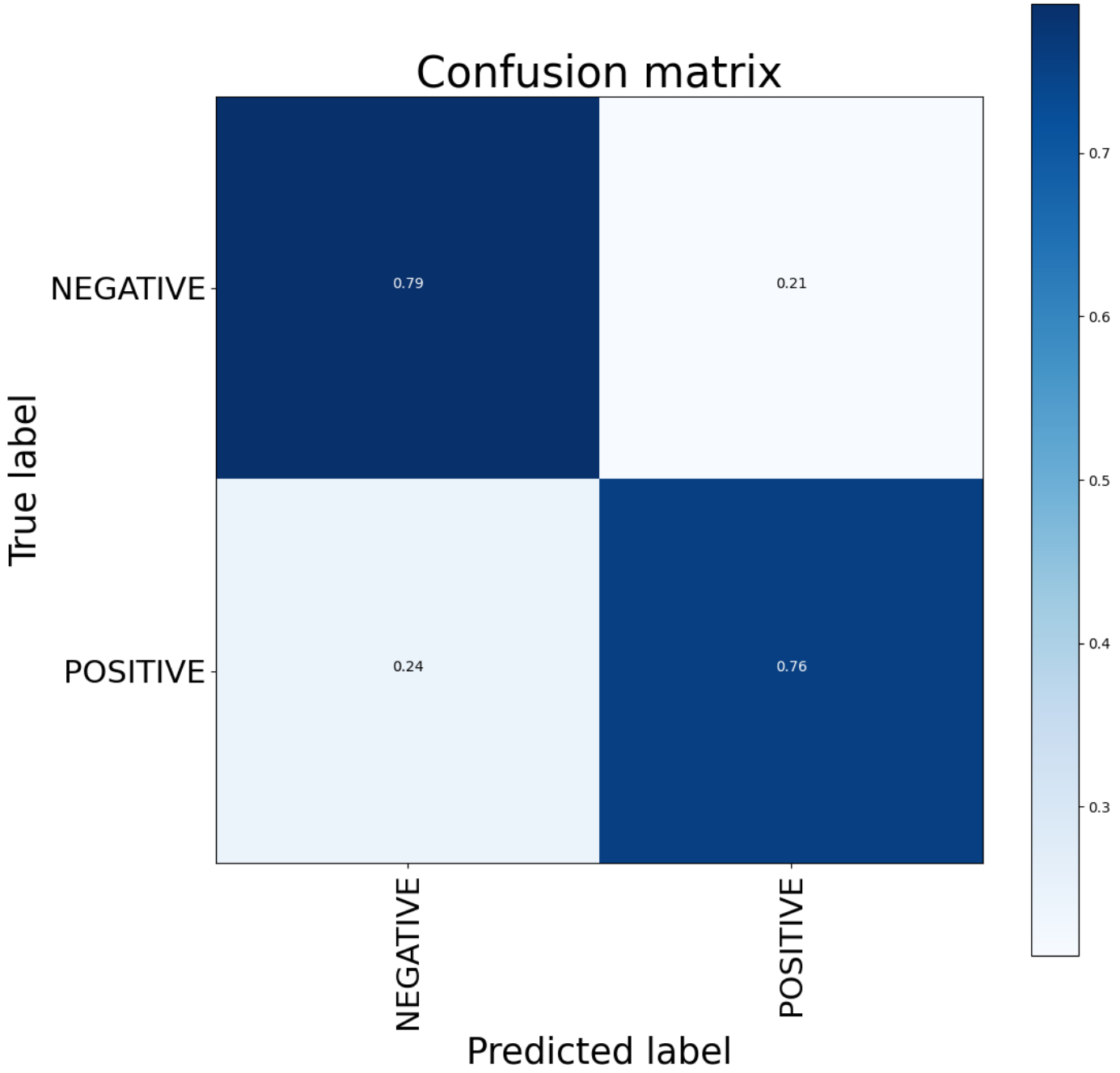
Out[ ]:
```
{'label': 'NEGATIVE',
 'score': 0.013639895245432854,
 'elapsed_time': 0.09506726264953613}
```

In [ ]:
```
%%time
y_pred_1d = []
y_test_1d = list(df_test.target)
scores = model_gru.predict(x_test, verbose=1, batch_size=8000)
y_pred_1d = [decode_sentiment(score, include_neutral=False) for score in scores]
```

```
40/40 [==============================] - 2s 35ms/step
CPU times: user 2.98 s, sys: 723 ms, total: 3.71 s
Wall time: 3.78 s
```

In [ ]:
```
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=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90, fontsize=22)
    plt.yticks(tick_marks, classes, fontsize=22)

    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=25)
    plt.xlabel('Predicted label', fontsize=25)
```

In [ ]:
```
%%time

cnf_matrix = confusion_matrix(y_test_1d, y_pred_1d)
plt.figure(figsize=(12,12))
plot_confusion_matrix(cnf_matrix, classes=df_train.target.unique(), title="Confusion mat
plt.show()
```




```
CPU times: user 1.93 s, sys: 231 ms, total: 2.17 s
Wall time: 2.5 s
```

In [ ]:
```
print(classification_report(y_test_1d, y_pred_1d))
```

```
              precision    recall  f1-score   support

    NEGATIVE       0.76      0.79      0.78    159986
    POSITIVE       0.78      0.76      0.77    160014

    accuracy                           0.77    320000
   macro avg       0.77      0.77      0.77    320000
weighted avg       0.77      0.77      0.77    320000
```

In [ ]:
```
accuracy_score(y_test_1d, y_pred_1d)
```

Out[ ]:
```
0.774125
```

In [ ]:
```
WORD2VEC_MODEL = "modeltest2_gru.txt"
model_gru.save(KERAS_MODEL)
w2v_model.save(WORD2VEC_MODEL)
```

```
pickle.dump(tokenizer, open(TOKENIZER_MODEL, "wb"), protocol=0)
pickle.dump(encoder, open(ENCODER_MODEL, "wb"), protocol=0)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: UserWarning:
You are saving your model as an HDF5 file via `model.save()`. This file format is consid
ered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_mo
del.keras')`.
  saving_api.save_model(
```

Bidirectional

In [ ]:
```
from keras.layers import Bidirectional, LSTM

model_bi = Sequential()
model_bi.add(Embedding(vocab_size, 2, input_length=SEQUENCE_LENGTH))
model_bi.add(Bidirectional(LSTM(32, return_sequences=False)))  # Use Bidirectional LSTM
model_bi.add(Dense(1, activation='sigmoid'))

model_bi.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 300, 2)            581422

 bidirectional (Bidirection  (None, 64)                8960
 al)

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 590447 (2.25 MB)
Trainable params: 590447 (2.25 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
model_bi.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

In [ ]:
```
history = model_bi.fit(x_train, y_train, batch_size=128, epochs=6, verbose=1, validation
```

```
Epoch 1/6
8000/8000 [==============================] - 2456s 306ms/step - loss: 0.4806 - accuracy:
0.7689 - val_loss: 0.4608 - val_accuracy: 0.7813
Epoch 2/6
7686/8000 [===========================>..] - ETA: 1:30 - loss: 0.4386 - accuracy: 0.7951
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-31-12e7c69a9eb2> in <cell line: 1>()
----> 1 history = model_bi.fit(x_train, y_train, batch_size=128, epochs=6, verbose=1, va
lidation_split=0.2)

/usr/local/lib/python3.10/dist-packages/keras/src/utils/traceback_utils.py in error_hand
ler(*args, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py in fit(self, x, y,
batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, clas
s_weight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_ba
tch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1740                         ):
   1741                             callbacks.on_train_batch_begin(step)
-> 1742                             tmp_logs = self.train_function(iterator)
   1743                             if data_handler.should_sync:
   1744                                 context.async_wait()

/usr/local/lib/python3.10/dist-packages/tensorflow/python/util/traceback_utils.py in err
or_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/polymorphic_function/pol
ymorphic_function.py in __call__(self, *args, **kwds)
    823
    824       with OptionalXlaContext(self._jit_compile):
--> 825         result = self._call(*args, **kwds)
    826
    827       new_tracing_count = self.experimental_get_tracing_count()

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/polymorphic_function/pol
ymorphic_function.py in _call(self, *args, **kwds)
    855       # In this case we have created variables on the first call, so we run the
    856       # defunned version which is guaranteed to never create variables.
--> 857       return self._no_variable_creation_fn(*args, **kwds)  # pylint: disable=not
-callable
    858     elif self._variable_creation_fn is not None:
    859       # Release the lock early so that multiple threads can perform the call

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/polymorphic_function/tra
cing_compiler.py in __call__(self, *args, **kwargs)
    146       (concrete_function,
    147        filtered_flat_args) = self._maybe_define_function(args, kwargs)
--> 148     return concrete_function._call_flat(
    149         filtered_flat_args, captured_inputs=concrete_function.captured_inputs)
# pylint: disable=protected-access
    150

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/polymorphic_function/mon
omorphic_function.py in _call_flat(self, args, captured_inputs)
   1347         and executing_eagerly):
   1348       # No tape is watching; skip to running the function.
-> 1349       return self._build_call_outputs(self._inference_function(*args))
   1350     forward_backward = self._select_forward_and_backward_functions(
```

```
   1351         args,

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/polymorphic_function/atomic_function.py in __call__(self, *args)
    194         with record.stop_recording():
    195           if self._bound_context.executing_eagerly():
--> 196             outputs = self._bound_context.call_function(
    197                 self.name,
    198                 list(args),

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/context.py in call_function(self, name, tensor_inputs, num_outputs)
   1455     cancellation_context = cancellation.context()
   1456     if cancellation_context is None:
-> 1457       outputs = execute.execute(
   1458           name.decode("utf-8"),
   1459           num_outputs=num_outputs,

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51   try:
     52     ctx.ensure_initialized()
---> 53     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                         inputs, attrs, num_outputs)
     55   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

In [ ]:
```
%%time
score = model_gru.evaluate(x_test, y_test, batch_size=BATCH_SIZE)
print()
print("ACCURACY:",score[1])
print("LOSS:",score[0])
```

In [ ]:
```
def plot_acc_loss(history):

    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)

    plt.plot(epochs, acc, 'bo', label = 'Training Accuracy')
    plt.plot(epochs, val_acc, 'r', label = 'Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.figure()
    plt.plot(epochs, loss, 'bo', label = 'Training Loss')
    plt.plot(epochs, val_loss, 'r', label = 'Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

In [ ]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

In [ ]:
```
def decode_sentiment(score, include_neutral=True):
    if include_neutral:
        label = NEUTRAL
        if score <= SENTIMENT_THRESHOLDS[0]:
            label = NEGATIVE
        elif score >= SENTIMENT_THRESHOLDS[1]:
            label = POSITIVE

        return label
    else:
        return NEGATIVE if score < 0.5 else POSITIVE
```

In [ ]:
```
def predict(text, include_neutral=True):
    start_at = time.time()
    # Tokenize text
    x_test = pad_sequences(tokenizer.texts_to_sequences([text]), maxlen=SEQUENCE_LENGTH)
    # Predict
    score = model_gru.predict([x_test])[0]
    # Decode sentiment
    label = decode_sentiment(score, include_neutral=include_neutral)

    return {"label": label, "score": float(score),
       "elapsed_time": time.time()-start_at}
```

In [ ]:
```
predict("I dont like to talk to anyone today.")
```

In [ ]:
```
%%time
y_pred_1d = []
y_test_1d = list(df_test.target)
scores = model_gru.predict(x_test, verbose=1, batch_size=8000)
y_pred_1d = [decode_sentiment(score, include_neutral=False) for score in scores]
```

In [ ]:
```
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=30)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90, fontsize=22)
    plt.yticks(tick_marks, classes, fontsize=22)
```

```
    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=25)
    plt.xlabel('Predicted label', fontsize=25)
```

In [ ]:
```
%%time

cnf_matrix = confusion_matrix(y_test_1d, y_pred_1d)
plt.figure(figsize=(12,12))
plot_confusion_matrix(cnf_matrix, classes=df_train.target.unique(), title="Confusion mat
plt.show()
```

In [ ]:
```
print(classification_report(y_test_1d, y_pred_1d))
```

In [ ]:
```
accuracy_score(y_test_1d, y_pred_1d)
```

In [ ]:
```
KERAS_MODEL = "modeltest2_bi.h5"
WORD2VEC_MODEL = "modeltest2_bi.pkl"
TOKENIZER_MODEL = "tokenizertest2_bi.pkl"
ENCODER_MODEL = "encodertest2_bi.pkl"

model_gru.save(KERAS_MODEL)
w2v_model.save(WORD2VEC_MODEL)
pickle.dump(tokenizer, open(TOKENIZER_MODEL, "wb"), protocol=0)
pickle.dump(encoder, open(ENCODER_MODEL, "wb"), protocol=0)
```