Sentiment Text Analysis with NLP Classifiers

This project involves the analysis of tweets about the coronavirus, with the goal of performing Sentiment Analysis using BERT and RoBERTa algorithms to predict the emotions of tweets (Positive, Negative, or Neutral).

Custom functions definition:

In [3]:
```
def plot_confusion_matrix(y_true, y_pred, title):
    fig, ax = plt.subplots(figsize=(5, 5))
    labels = ['Negative', 'Neutral', 'Positive']
    confusion_mat = confusion_matrix(y_true, y_pred)
    ax = sns.heatmap(confusion_mat, annot=True, cmap="Blues", fmt='g', cbar=False, anr
    plt.title(title, fontsize=20)
    ax.xaxis.set_ticklabels(labels, fontsize=17)
    ax.yaxis.set_ticklabels(labels, fontsize=17)
    ax.set_ylabel('Actual', fontsize=20)
    ax.set_xlabel('Predicted', fontsize=20)
    plt.show()
```

Loading the data

In [4]:
```
df = pd.read_csv('input/covid-19-nlp-text-classification/Corona_NLP_train.csv',encodir
df_test = pd.read_csv('input/covid-19-nlp-text-classification/Corona_NLP_test.csv')
```

Checking the data

In [5]:
```
df.head()
```

Out[5]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41157 entries, 0 to 41156
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   UserName       41157 non-null  int64 
 1   ScreenName     41157 non-null  int64 
 2   Location       32567 non-null  object
 3   TweetAt        41157 non-null  object
 4   OriginalTweet  41157 non-null  object
 5   Sentiment      41157 non-null  object
dtypes: int64(2), object(4)
memory usage: 1.9+ MB
```

Convert the date column 'TweetAt' to pandas datetime format to improve its usability in the further analysis.

In [7]:
```
df['TweetAt'] = pd.to_datetime(df['TweetAt'])
```

In [8]:
```
df.head()
```

Out[8]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	2020-03-16	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	2020-03-16	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	2020-03-16	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	2020-03-16	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	2020-03-16	Me, ready to go at supermarket during the #COV...	Extremely Negative

Handling the Duplicate Data

In [9]:
```
df.drop_duplicates(subset='OriginalTweet',inplace=True)
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41157 entries, 0 to 41156
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   UserName       41157 non-null  int64         
 1   ScreenName     41157 non-null  int64         
 2   Location       32567 non-null  object        
 3   TweetAt        41157 non-null  datetime64[ns]
 4   OriginalTweet  41157 non-null  object        
 5   Sentiment      41157 non-null  object        
dtypes: datetime64[ns](1), int64(2), object(3)
memory usage: 2.2+ MB
```

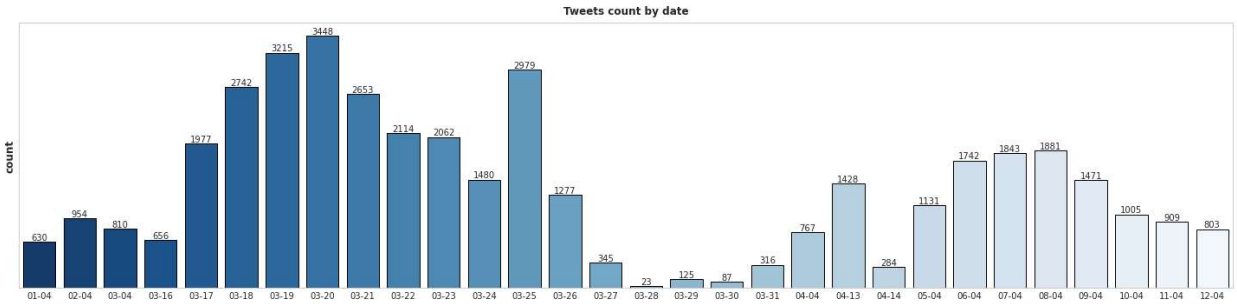
No duplicate data founded.

EDA

Data Count by Date

In [11]:
```
tweets_per_day = df['TweetAt'].dt.strftime('%m-%d').value_counts().sort_index().reset_
```

In [12]:
```
plt.figure(figsize=(20,5))
ax = sns.barplot(x='index', y='counts', data=tweets_per_day,edgecolor = 'black',ci=Fal
plt.title('Tweets count by date')
plt.yticks([])
ax.bar_label(ax.containers[0])
plt.ylabel('count')
plt.xlabel('')
plt.show()
```

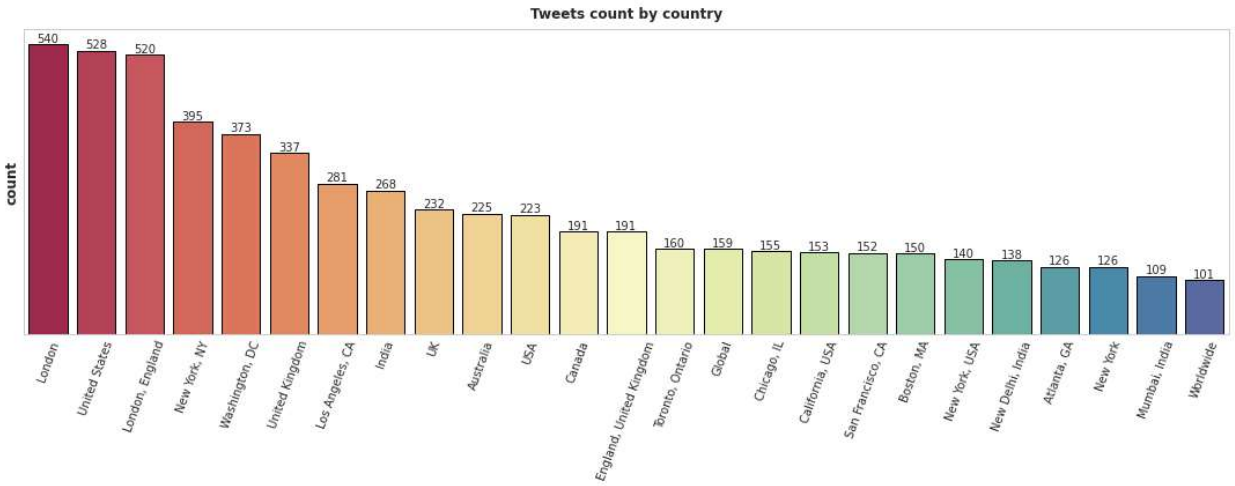


In the dataset, there are some days without tweets. Among the days with tweets, most of them were made around the end of March, from the 18th of March to the 26th of March.

Tweets data per country and city

In [13]:
```
tweets_per_country = df['Location'].value_counts().loc[lambda x : x > 100].reset_index
```

In [14]:
```
plt.figure(figsize=(15,6))
ax = sns.barplot(x='index', y='counts', data=tweets_per_country,edgecolor = 'black',ci
plt.title('Tweets count by country')
plt.xticks(rotation=70)
plt.yticks([])
ax.bar_label(ax.containers[0])
plt.ylabel('count')
plt.xlabel('')
plt.show()
```



Tweets Data Preprocessing

In the following, perform some data cleaning on the raw text of the tweets. To simplify the analysis, only keep the columns 'Originaltweet' (raw tweets) and the target column 'Sentiment'.

In [15]:
```
df = df[['OriginalTweet','Sentiment']]
```

In [16]:
```
df_test = df_test[['OriginalTweet','Sentiment']]
```

Define custom functions to clean the text of the tweets.

In [17]:
```
##CUSTOM DEFINED FUNCTIONS TO CLEAN THE TWEETS

#Clean emojis from text
def strip_emoji(text):
    return re.sub(emoji.get_emoji_regexp(), r"", text) #remove emoji

#Remove punctuations, links, mentions and \r\n new line characters
def strip_all_entities(text):
    text = text.replace('\r', '').replace('\n', ' ').replace('\n', ' ').lower() #remov
    text = re.sub(r"(?:\@|https?\://)\S+", "", text) #remove links and mentions
    text = re.sub(r'[^\x00-\x7f]',r'', text) #remove non utf8/ascii characters such as
    banned_list= string.punctuation + 'Ã'+'±'+'ã'+'¼'+'â'+'»'+'§'
    table = str.maketrans('', '', banned_list)
    text = text.translate(table)
    return text

#clean hashtags at the end of the sentence, and keep those in the middle of the senter
def clean_hashtags(tweet):
    new_tweet = " ".join(word.strip() for word in re.split('#(?!(?:hashtag)\b)[\w-]+(
    new_tweet2 = " ".join(word.strip() for word in re.split('#|_', new_tweet)) #remove
    return new_tweet2

#Filter special characters such as & and $ present in some words
def filter_chars(a):
    sent = []
    for word in a.split(' '):
        if ('$' in word) | ('&' in word):
            sent.append('')
        else:
            sent.append(word)
    return ' '.join(sent)
```

```
def remove_mult_spaces(text): # remove multiple spaces
    return re.sub("\s\s+" , " ", text)
```

In [18]:
```
texts_new = []
for t in df.OriginalTweet:
    texts_new.append(remove_mult_spaces(filter_chars(clean_hashtags(strip_all_entities
```

In [19]:
```
texts_new_test = []
for t in df_test.OriginalTweet:
    texts_new_test.append(remove_mult_spaces(filter_chars(clean_hashtags(strip_all_ent
```

Create a new column in both the train and test sets to store the cleaned versions of the tweets' text.

In [20]:
```
df['text_clean'] = texts_new
df_test['text_clean'] = texts_new_test
```

In [21]:
```
df['text_clean'].head()
```

Out[21]:
```
0                                                and and
1    advice talk to your neighbours family to excha...
2    coronavirus australia woolworths to give elder...
3    my food stock is not the only one which is emp...
4    me ready to go at supermarket during the covid...
Name: text_clean, dtype: object
```

In [22]:
```
df_test['text_clean'].head()
```

Out[22]:
```
0    trending new yorkers encounter empty supermark...
1    when i couldnt find hand sanitizer at fred mey...
2    find out how you can protect yourself and love...
3    panic buying hits newyork city as anxious shop...
4    toiletpaper dunnypaper coronavirus coronavirus...
Name: text_clean, dtype: object
```

In [23]:
```
df['text_clean'][1:8].values
```

Out[23]:
```
array(['advice talk to your neighbours family to exchange phone numbers create contac
t list with phone numbers of neighbours schools employer chemist gp set up online sho
pping accounts if poss adequate supplies of regular meds but not over order',
       'coronavirus australia woolworths to give elderly disabled dedicated shopping
hours amid covid19 outbreak',
       'my food stock is not the only one which is empty please dont panic there will
be enough food for everyone if you do not take more than you need stay calm stay safe
covid19france covid19 covid19 coronavirus confinement confinementotal confinementgene
ral',
       'me ready to go at supermarket during the covid19 outbreak not because im para
noid but because my food stock is litteraly empty the coronavirus is a serious thing
but please dont panic it causes shortage coronavirusfrance restezchezvous stayathome
confinement',
       'as news of the regions first confirmed covid19 case came out of sullivan coun
ty last week people flocked to area stores to purchase cleaning supplies hand sanitiz
er food toilet paper and other goods reports',
       'cashier at grocery store was sharing his insights on covid19 to prove his cre
dibility he commented im in civics class so i know what im talking about',
       'was at the supermarket today didnt buy toilet paper rebel toiletpapercrisis c
ovid19'],
      dtype=object)
```

Create a column to host the length of the cleaned text to check if cleaning the text removed too much or almost the entire tweet.

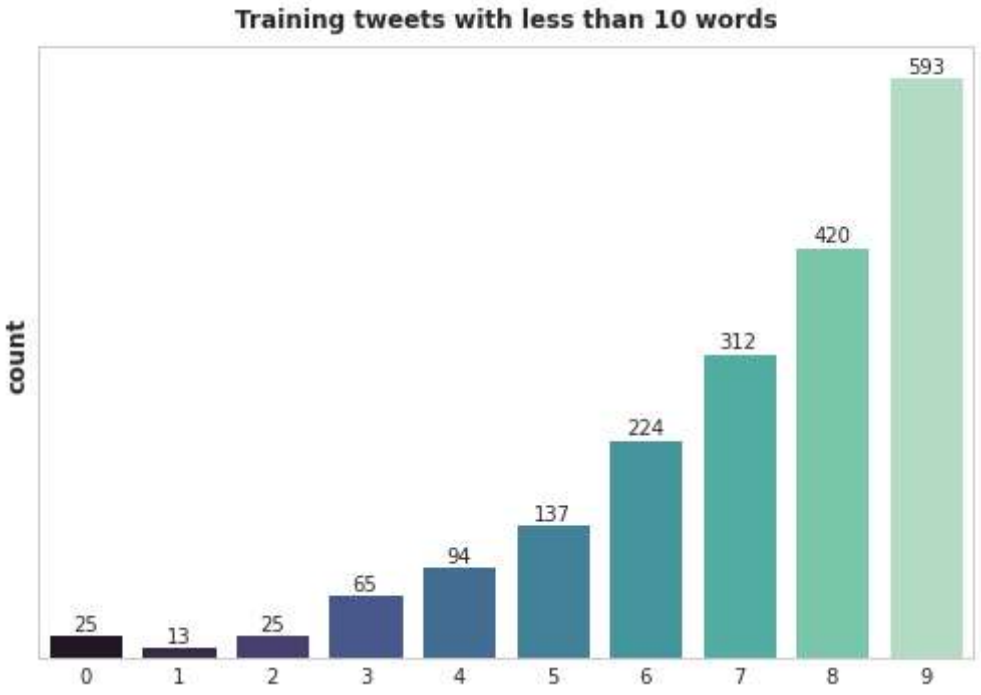
In [24]:
```
text_len = []
for text in df.text_clean:
    tweet_len = len(text.split())
    text_len.append(tweet_len)
```

In [25]:
```
df['text_len'] = text_len
```

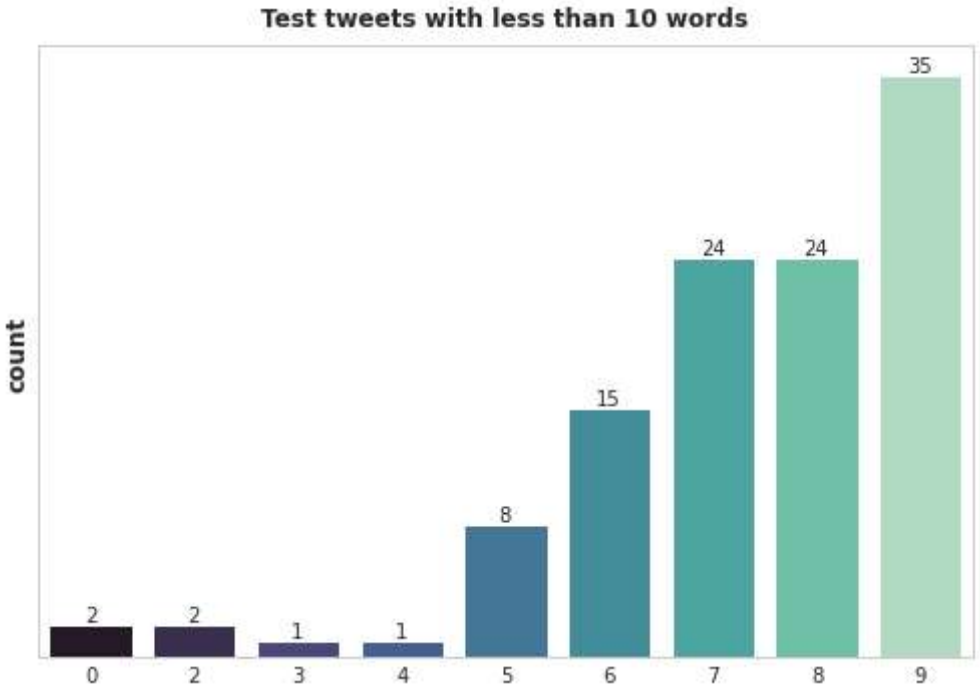
In [26]:
```
text_len_test = []
for text in df_test.text_clean:
    tweet_len = len(text.split())
    text_len_test.append(tweet_len)
```

In [27]:
```
df_test['text_len'] = text_len_test
```

In [28]:
```
plt.figure(figsize=(7,5))
ax = sns.countplot(x='text_len', data=df[df['text_len']<10], palette='mako')
plt.title('Training tweets with less than 10 words')
plt.yticks([])
ax.bar_label(ax.containers[0])
plt.ylabel('count')
plt.xlabel('')
plt.show()
```



In [29]:
```
plt.figure(figsize=(7,5))
ax = sns.countplot(x='text_len', data=df_test[df_test['text_len']<10], palette='mako')
plt.title('Test tweets with less than 10 words')
plt.yticks([])
ax.bar_label(ax.containers[0])
plt.ylabel('count')
plt.xlabel('')
plt.show()
```



There are many cleaned tweets with 0 words, which is a result of the cleaning performed earlier. This indicates that some tweets contained only mentions, hashtags, and links, which have been removed. Drop these empty tweets as well as those with fewer than 5 words.

In [30]:
```
print(f" DF SHAPE: {df.shape}")
print(f" DF TEST SHAPE: {df_test.shape}")
```

```
 DF SHAPE: (41157, 4)
 DF TEST SHAPE: (3798, 4)
```

In [31]:
```
df = df[df['text_len'] > 4]
```

In [32]:
```
df_test = df_test[df_test['text_len'] > 4]
```

In [33]:
```
print(f" DF SHAPE: {df.shape}")
print(f" DF TEST SHAPE: {df_test.shape}")
```

```
 DF SHAPE: (40935, 4)
 DF TEST SHAPE: (3792, 4)
```

Training data preprocessing

Checking the tokenizer version of the sentences.

Import the BERT tokenizer.

In [34]:
```
tokenizer = BertTokenizerFast.from_pretrained('bert-base-uncased')
```

```
Downloading:   0%|          | 0.00/226k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/455k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/28.0 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

In [35]:
```
token_lens = []

for txt in df['text_clean'].values:
```

```
    tokens = tokenizer.encode(txt, max_length=512, truncation=True)
    token_lens.append(len(tokens))

max_len=np.max(token_lens)
```

In [36]:
```
print(f"MAX TOKENIZED SENTENCE LENGTH: {max_len}")
```

```
MAX TOKENIZED SENTENCE LENGTH: 100
```

Checking the long tokenized sentences (with more than 80 tokens):

In [37]:
```
token_lens = []

for i,txt in enumerate(df['text_clean'].values):
    tokens = tokenizer.encode(txt, max_length=512, truncation=True)
    token_lens.append(len(tokens))
    if len(tokens)>80:
        print(f"INDEX: {i}, TEXT: {txt}")
```

```
INDEX: 1622, TEXT: zsah policie proti spekulantm s roukami na mj popud hejtman steckh
o kraje ve spoluprci s podle krizovho zkona zajistil tm 700 tisrouek od firmy kter je
mla dodat na zdravotnkm ale na posledn chvli se snaila navyovat cenu spolutozvladneme
INDEX: 13623, TEXT: hoy aplaudo a mi segunda familia aplaudoanuestrosheroes aquellos
con los que he compartido tantas noches de trabajo y tanta alegra s que como siempre
dan todo por el bien de su comunidad presidente por ellos tambin cuarentenanacionalya
cuidemosalosquecuidan
INDEX: 16548, TEXT: bir mddettir spermarketlerin lojistik hizmeti avusturya ordusu de
steiyle yaplyor dn corona tedavisi iin 22 milyon luk bir aratrma gelitirme btesi akla
d hkmet geen hafta da 35 milyon luk 2 yardm paketi aklanmt viyanadan haberler bu kada
r
INDEX: 36953, TEXT: 11 nisan cumartesi itibariyle bbnin tm hizmetleri sokaa kma serbe
stisi olanlar iin devam edecek halk ekmek ve hamidiye su 100 retime geti bb tm stanbu
lun gda ihtiyacna yetecek kapasitededir halkmz sakin olsun ve gvende hissetsin ltfen
herkes evine dnsn
```

These sentences are not in english. They should be dropped.

In [38]:
```
df['token_lens'] = token_lens
```

In [39]:
```
df = df.sort_values(by='token_lens', ascending=False)
df.head(20)
```

Out[39]:

	OriginalTweet	Sentiment	text_clean	text_len	token_lens
1638	ZÃ¡sah policie proti spekulant?m s rouÂkami. ...	Neutral	zsah policie proti spekulantm s roukami na mj ...	39	100
37156	11 Nisan Cumartesi itibariyle ?BBÂnin tÃ¼m hi...	Neutral	11 nisan cumartesi itibariyle bbnin tm hizmetl...	39	98
16632	Bir mÃ¼ddettir sÃ¼permarketlerin lojistik hizm...	Neutral	bir mddettir spermarketlerin lojistik hizmeti ...	36	92
13691	Hoy aplaudo a mi segunda familia #AplaudoANues...	Neutral	hoy aplaudo a mi segunda familia aplaudoanuest...	38	84
27005	Supermercados Econo confirman que un empleado ...	Neutral	supermercados econo confirman que un empleado ...	39	80
14593	Na, wer war denn da am Werk? Gestern Nachmitta...	Extremely Negative	na wer war denn da am werk gestern nachmittag ...	37	80
28899	Kindly contact Us bamy global merchandise for ...	Positive	kindly contact us bamy global merchandise for ...	37	80
11213	Keine WertgegenstÃ¤nde im Fahrzeug lassen! - D...	Negative	keine wertgegenstnde im fahrzeug lassen diesen...	33	79
4844	Impct of #coronavirus i hve sen hw civilizd pp...	Extremely Negative	impct of coronavirus i hve sen hw civilizd ppl...	48	79
18913	#CroozefmNews \r\r\nPresident Museveni has ord...	Extremely Negative	croozefmnews president museveni has ordered th...	35	79
30206	#LDA City Lahore Residential Files Prices Upda...	Neutral	lda city lahore residential files prices updat...	43	78
26678	Eine wahre #CoronaGeschichte:\r\r\n\r\r\nWenn ...	Neutral	eine wahre coronageschichte wenn dir an der su...	29	78
12389	Okay, so I just checked the drug prices for #P...	Positive	okay so i just checked the drug prices for pla...	35	77
1697	I work at a grocery store.\r\r\nWe wont get an...	Positive	i work at a grocery store we wont get any toil...	37	77
8730	?Bitte anschauen! (1/2)\r\r\n\r\r\nEmotionaler...	Negative	bitte anschauen 12 emotionaler aufruf von geha...	36	77
14582	hiked prices in the face of the Covid-19 crise...	Negative	hiked prices in the face of the covid19	47	77

	OriginalTweet	Sentiment	text_clean	text_len	token_lens
			crises...		
36305	Sterile disposable anti bacterial wet wipes an...	Negative	sterile disposable anti bacterial wet wipes an...	31	76
36306	For sell Sterile disposable anti bacterial wet...	Negative	for sell sterile disposable anti bacterial wet...	32	75
9238	Hi @Zomato I felt hungry loggd in ur app tryd ...	Extremely Positive	hi i felt hungry loggd in ur app tryd to add 4...	55	75
40778	#Covid_19 2days Stay-at-home activities\r\r\nU...	Neutral	covid19 2days stayathome activities up 630am f...	37	75

In [40]:
```
df = df.iloc[12:]
df.head()
```

Out[40]:

	OriginalTweet	Sentiment	text_clean	text_len	token_lens
12389	Okay, so I just checked the drug prices for #P...	Positive	okay so i just checked the drug prices for pla...	35	77
1697	I work at a grocery store.\r\r\nWe wont get an...	Positive	i work at a grocery store we wont get any toil...	37	77
8730	?Bitte anschauen! (1/2)\r\r\n\r\r\nEmotionaler...	Negative	bitte anschauen 12 emotionaler aufruf von geha...	36	77
14582	hiked prices in the face of the Covid-19 crise...	Negative	hiked prices in the face of the covid19 crises...	47	77
36305	Sterile disposable anti bacterial wet wipes an...	Negative	sterile disposable anti bacterial wet wipes an...	31	76

The dataset looks more clean now. Shuffle the new dataset and reset the index.

In [41]:
```
df = df.sample(frac=1).reset_index(drop=True)
```

Test data preprocessing

Performing the data cleaning based on the tokenized sentences on the test set.

In [42]:
```
token_lens_test = []

for txt in df_test['text_clean'].values:
    tokens = tokenizer.encode(txt, max_length=512, truncation=True)
    token_lens_test.append(len(tokens))

max_len=np.max(token_lens_test)
```

In [43]:
```
print(f"MAX TOKENIZED SENTENCE LENGTH: {max_len}")
```

```
MAX TOKENIZED SENTENCE LENGTH: 96
```

In [44]:
```
token_lens_test = []

for i,txt in enumerate(df_test['text_clean'].values):
    tokens = tokenizer.encode(txt, max_length=512, truncation=True)
    token_lens_test.append(len(tokens))
    if len(tokens)>80:
        print(f"INDEX: {i}, TEXT: {txt}")
```

```
INDEX: 286, TEXT: so hard to decide as much as i want to hodl my 2 ccdcv4 token our p
lace is declared to lock down due to covid19 i will use this to buy some food to stoc
k txnid093bd1db0c0d3a62af15883138a5f57d4cef35ae14e31e602b74489dd2524c7f my b
INDEX: 345, TEXT: informoval jsem zstupce vech obchodnch etzc o aktulnch opatench vld
y etzce jsou zsobovny na 95 take nen dvod panikait zsoby potravin fakt nedojdou nen o
pravdu dvod dnes obsadit a vykoupit supermarkety
INDEX: 2380, TEXT: ahora seguid llorando por el papel higinico que no he comprado por
que an tengo seguid creando histeria y preocupacin poniendo fotos de gente en pnico y
estanteras vacas que yo seguir yendo a comercios responsables de barrio donde nos cui
dan hoy y siempre gracias
```

In [45]:
```
df_test['token_lens'] = token_lens_test
```

In [46]:
```
df_test = df_test.sort_values(by='token_lens', ascending=False)
df_test.head(10)
```

Out[46]:

	OriginalTweet	Sentiment	text_clean	text_len	token_lens
286	@Rhett800cc So hard to decide??. As much as I ...	Negative	so hard to decide as much as i want to hodl my...	38	96
2383	Ahora seguid llorando por el papel higiénico (...	Negative	ahora seguid llorando por el papel higinico qu...	44	94
345	Informoval jsem zástupce v□ech obchodních ?et?...	Neutral	informoval jsem zstupce vech obchodnch etzc o ...	31	86
1485	DTF-Don□t Touch Face\r\r\nDWBH-Do Wash Both Ha...	Extremely Negative	dtfdont touch face dwbhdo wash both hands gtfo...	42	77
1209	I□m in the DC/Maryland/Virginia (DMV) area &am...	Positive	im in the dcmarylandvirginia dmv area amphave ...	45	74
3505	Stop misusing ur privilege amp grow up Some1 c...	Positive	stop misusing ur privilege amp grow up some1 c...	57	73
1789	For those that are cashlong, patient,calm&...	Extremely Positive	for those that are cashlong patientcalmamphave...	44	71
855	Lidl is total chaos, queues as long as the ais...	Extremely Negative	lidl is total chaos queues as long as the aisl...	62	70
2740	COVID-19: Your government will save ITSELF not...	Positive	covid19 your government will save itself not y...	43	70
2997	Stop #frenzybuying. You don't need most of wha...	Extremely Negative	stop frenzybuying you dont need most of what y...	38	70

In [47]:
```
df_test = df_test.iloc[5:]
df_test.head(3)
```

Out[47]:

	OriginalTweet	Sentiment	text_clean	text_len	token_lens
3505	Stop misusing ur privilege amp grow up Some1 c...	Positive	stop misusing ur privilege amp grow up some1 c...	57	73
1789	For those that are cashlong, patient,calm&...	Extremely Positive	for those that are cashlong patientcalmamphave...	44	71
855	Lidl is total chaos, queues as long as the ais...	Extremely Negative	lidl is total chaos queues as long as the aisl...	62	70

In [48]:
```
df_test = df_test.sample(frac=1).reset_index(drop=True)
```

Sentiment data column analysis

Looking at the target column 'Sentiment'.

In [49]:
```
df['Sentiment'].value_counts()
```

Out[49]:
```
Positive              11381
Negative               9889
Neutral                7560
Extremely Positive     6618
Extremely Negative     5475
Name: Sentiment, dtype: int64
```

The first thing we can do is encode the categories with numbers and then create three possible emotions: Positive, Neutral, and Negative.

In [50]:
```
df['Sentiment'] = df['Sentiment'].map({'Extremely Negative':0,'Negative':0,'Neutral':1
```

In [51]:
```
df_test['Sentiment'] = df_test['Sentiment'].map({'Extremely Negative':0,'Negative':0,'
```

In [52]:
```
df['Sentiment'].value_counts()
```

Out[52]:
```
2    17999
0    15364
1     7560
Name: Sentiment, dtype: int64
```

Notice that the three classes are imbalanced. Then proceed with oversampling the training set to remove bias towards the majority classes.

Class Balancing by RandomOverSampler

In [53]:
```
ros = RandomOverSampler()
train_x, train_y = ros.fit_resample(np.array(df['text_clean']).reshape(-1, 1), np.arra
train_os = pd.DataFrame(list(zip([x[0] for x in train_x], train_y)), columns = ['text_
```

In [54]:
```
train_os['Sentiment'].value_counts()
```

Out[54]:
```
2    17999
1    17999
0    17999
Name: Sentiment, dtype: int64
```

Train - Validation - Test split

In [55]:
```
X = train_os['text_clean'].values
y = train_os['Sentiment'].values
```

We will extract a validation set from the training data to monitor validation accuracy and prevent overfitting.

In [56]:
```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.1, stratify=y,
```

In [57]:
```
X_test = df_test['text_clean'].values
y_test = df_test['Sentiment'].values
```

One Hot Encoding

In [58]:
```
y_train_le = y_train.copy()
y_valid_le = y_valid.copy()
y_test_le = y_test.copy()
```

In [59]:
```
ohe = preprocessing.OneHotEncoder()
y_train = ohe.fit_transform(np.array(y_train).reshape(-1, 1)).toarray()
y_valid = ohe.fit_transform(np.array(y_valid).reshape(-1, 1)).toarray()
y_test = ohe.fit_transform(np.array(y_test).reshape(-1, 1)).toarray()
```

In [60]:
```
print(f"TRAINING DATA: {X_train.shape[0]}\nVALIDATION DATA: {X_valid.shape[0]}\nTESTIN
```

```
TRAINING DATA: 48597
VALIDATION DATA: 5400
TESTING DATA: 3787
```

Baseline model: Naive Bayes Classifier

Before implementing BERT algorithms, define a simple Naive Bayes baseline model to classify the tweets.

First, tokenize the tweets using CountVectorizer.

In [61]:
```
clf = CountVectorizer()
X_train_cv =  clf.fit_transform(X_train)
X_test_cv = clf.transform(X_test)
```

Then create the TF-IDF (term-frequency times inverse document-frequency) versions of the tokenized tweets.

In [62]:
```
tf_transformer = TfidfTransformer(use_idf=True).fit(X_train_cv)
X_train_tf = tf_transformer.transform(X_train_cv)
X_test_tf = tf_transformer.transform(X_test_cv)
```

Next, define the Naive Bayes Classifier model

In [63]:
```
nb_clf = MultinomialNB()
```

In [64]:
```
nb_clf.fit(X_train_tf, y_train_le)
```

Out[64]:
```
MultinomialNB()
```

In [65]:
```
nb_pred = nb_clf.predict(X_test_tf)
```

In [66]:
```
print('\tClassification Report for Naive Bayes:\n\n',classification_report(y_test_le,r
```

```
        Classification Report for Naive Bayes:

              precision    recall  f1-score   support

    Negative       0.71      0.79      0.75      1629
     Neutral       0.60      0.45      0.51       614
    Positive       0.75      0.73      0.74      1544

    accuracy                           0.71      3787
   macro avg       0.68      0.66      0.66      3787
weighted avg       0.70      0.71      0.70      3787
```

The algorithm's performance is decent. The F1 score is approximately 75% for the more populated classes (Negative and Positive emotions) but lower for the Neutral class (F1 = 51%). Overall, the accuracy is at 70%.

In the next section we will perform the sentiment analysis using BERT algorithms.

Sentiment Test Data Analysis Using NLP Algorithms

Define a custom tokenizer function and call the encode_plus method of the BERT tokenizer.

In [67]:
```
MAX_LEN=128
```

In [68]:
```
def tokenize(data,max_len=MAX_LEN) :
    input_ids = []
    attention_masks = []
    for i in range(len(data)):
        encoded = tokenizer.encode_plus(
            data[i],
            add_special_tokens=True,
            max_length=MAX_LEN,
            padding='max_length',
            return_attention_mask=True
        )
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])
    return np.array(input_ids),np.array(attention_masks)
```

Then, apply the tokenizer function to the train, validation and test datasets.

In [69]:
```
train_input_ids, train_attention_masks = tokenize(X_train, MAX_LEN)
val_input_ids, val_attention_masks = tokenize(X_valid, MAX_LEN)
```

```
test_input_ids, test_attention_masks = tokenize(X_test, MAX_LEN)
```

NLP Algorithms: BERT

Import the BERT model from the pretrained library from Hugging face.

In [70]:
```
bert_model = TFBertModel.from_pretrained('bert-base-uncased')
```

```
Downloading:   0%|          | 0.00/511M [00:00<?, ?B/s]
Some layers from the model checkpoint at bert-base-uncased were not used when initial
izing TFBertModel: ['mlm___cls', 'nsp___cls']
- This IS expected if you are initializing TFBertModel from the checkpoint of a model
trained on another task or with another architecture (e.g. initializing a BertForSequ
enceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertModel from the checkpoint of a m
odel that you expect to be exactly identical (initializing a BertForSequenceClassific
ation model from a BertForSequenceClassification model).
All the layers of TFBertModel were initialized from the model checkpoint at bert-base
-uncased.
If your task is similar to the task the model of the checkpoint was trained on, you c
an already use TFBertModel for predictions without further training.
```

Create a custom function to host the pre-trained BERT model and attach a 3-neuron output layer to it. This output layer is necessary for classifying the three different emotions in the dataset.

In [71]:
```
def create_model(bert_model, max_len=MAX_LEN):

    ##params###
    opt = tf.keras.optimizers.Adam(learning_rate=1e-5, decay=1e-7)
    loss = tf.keras.losses.CategoricalCrossentropy()
    accuracy = tf.keras.metrics.CategoricalAccuracy()


    input_ids = tf.keras.Input(shape=(max_len,),dtype='int32')

    attention_masks = tf.keras.Input(shape=(max_len,),dtype='int32')

    embeddings = bert_model([input_ids,attention_masks])[1]

    output = tf.keras.layers.Dense(3, activation="softmax")(embeddings)

    model = tf.keras.models.Model(inputs = [input_ids,attention_masks], outputs = outp

    model.compile(opt, loss=loss, metrics=accuracy)


    return model
```

In [72]:
```
model = create_model(bert_model, MAX_LEN)
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_1 (InputLayer)            [(None, 128)]        0
__________________________________________________________________________________________
input_2 (InputLayer)            [(None, 128)]        0
__________________________________________________________________________________________
tf_bert_model (TFBertModel)     TFBaseModelOutputWit 109482240   input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________
dense (Dense)                   (None, 3)            2307        tf_bert_model[0][1]
==========================================================================================
Total params: 109,484,547
Trainable params: 109,484,547
Non-trainable params: 0
__________________________________________________________________________________________
```

Start fine tuning the BERT Algorithms

In [73]:
```
history_bert = model.fit([train_input_ids,train_attention_masks], y_train, validation_
```

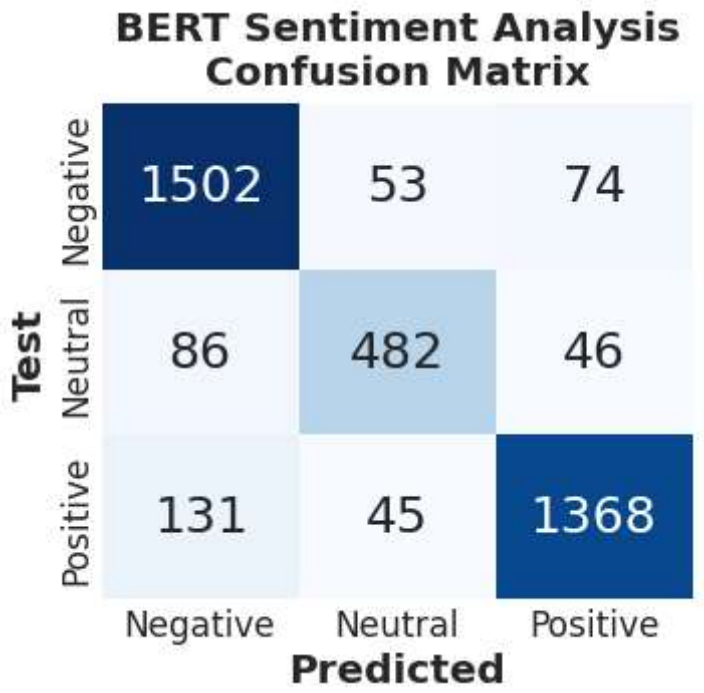
```
Epoch 1/4
1519/1519 [==============================] - 755s 489ms/step - loss: 0.5543 - categor
ical_accuracy: 0.7781 - val_loss: 0.3675 - val_categorical_accuracy: 0.8724
Epoch 2/4
1519/1519 [==============================] - 740s 487ms/step - loss: 0.2890 - categor
ical_accuracy: 0.8989 - val_loss: 0.2646 - val_categorical_accuracy: 0.9113
Epoch 3/4
1519/1519 [==============================] - 752s 495ms/step - loss: 0.1927 - categor
ical_accuracy: 0.9335 - val_loss: 0.2270 - val_categorical_accuracy: 0.9274
Epoch 4/4
1519/1519 [==============================] - 739s 487ms/step - loss: 0.1309 - categor
ical_accuracy: 0.9561 - val_loss: 0.2257 - val_categorical_accuracy: 0.9354
```

The results

In [74]:
```
result_bert = model.predict([test_input_ids,test_attention_masks])
```

In [76]:
```
y_pred_bert =  np.zeros_like(result_bert)
y_pred_bert[np.arange(len(y_pred_bert)), result_bert.argmax(1)] = 1
```

In [77]:
```
conf_matrix(y_test.argmax(1), y_pred_bert.argmax(1),'BERT Sentiment Analysis\nConfusio
```



In [78]:
```
print('\tClassification Report for BERT:\n\n',classification_report(y_test,y_pred_bert
```

```
        Classification Report for BERT:

              precision    recall  f1-score   support

    Negative       0.87      0.92      0.90      1629
     Neutral       0.83      0.79      0.81       614
    Positive       0.92      0.89      0.90      1544

   micro avg       0.89      0.89      0.89      3787
   macro avg       0.87      0.86      0.87      3787
weighted avg       0.89      0.89      0.88      3787
 samples avg       0.89      0.89      0.89      3787
```

NLP Algorithms: RoBERTa

In [79]:
```
tokenizer_roberta = RobertaTokenizerFast.from_pretrained("roberta-base")
```

```
Downloading:   0%|          | 0.00/878k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/446k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/1.29M [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/481 [00:00<?, ?B/s]
```

First, Check the length of the longest tokenized sentence by roberta tokenizer:

In [80]:
```
token_lens = []

for txt in X_train:
    tokens = tokenizer_roberta.encode(txt, max_length=512, truncation=True)
    token_lens.append(len(tokens))
max_length=np.max(token_lens)
max_length
```

Out[80]: 89

In [81]:
```
MAX_LEN=128
```

Then define the tokenization function

In [82]:
```
def tokenize_roberta(data,max_len=MAX_LEN) :
    input_ids = []
    attention_masks = []
    for i in range(len(data)):
        encoded = tokenizer_roberta.encode_plus(
            data[i],
            add_special_tokens=True,
            max_length=max_len,
            padding='max_length',
            return_attention_mask=True
        )
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])
    return np.array(input_ids),np.array(attention_masks)
```

In [83]:
```
train_input_ids, train_attention_masks = tokenize_roberta(X_train, MAX_LEN)
val_input_ids, val_attention_masks = tokenize_roberta(X_valid, MAX_LEN)
test_input_ids, test_attention_masks = tokenize_roberta(X_test, MAX_LEN)
```

RoBERTa Algorithms modeling

In [84]:
```
def create_model(bert_model, max_len=MAX_LEN):

    opt = tf.keras.optimizers.Adam(learning_rate=1e-5, decay=1e-7)
    loss = tf.keras.losses.CategoricalCrossentropy()
    accuracy = tf.keras.metrics.CategoricalAccuracy()

    input_ids = tf.keras.Input(shape=(max_len,),dtype='int32')
    attention_masks = tf.keras.Input(shape=(max_len,),dtype='int32')
    output = bert_model([input_ids,attention_masks])
    output = output[1]
    output = tf.keras.layers.Dense(3, activation=tf.nn.softmax)(output)
    model = tf.keras.models.Model(inputs = [input_ids,attention_masks],outputs = outpu
    model.compile(opt, loss=loss, metrics=accuracy)
    return model
```

In [85]:
```
roberta_model = TFRobertaModel.from_pretrained('roberta-base')
```

```
Downloading:   0%|          | 0.00/627M [00:00<?, ?B/s]
```

```
Some layers from the model checkpoint at roberta-base were not used when initializing
TFRobertaModel: ['lm_head']
- This IS expected if you are initializing TFRobertaModel from the checkpoint of a mo
del trained on another task or with another architecture (e.g. initializing a BertFor
SequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFRobertaModel from the checkpoint of
a model that you expect to be exactly identical (initializing a BertForSequenceClassi
fication model from a BertForSequenceClassification model).
All the layers of TFRobertaModel were initialized from the model checkpoint at robert
a-base.
If your task is similar to the task the model of the checkpoint was trained on, you c
an already use TFRobertaModel for predictions without further training.
```

In [86]:
```
model = create_model(roberta_model, MAX_LEN)
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_3 (InputLayer)            [(None, 128)]        0
__________________________________________________________________________________
input_4 (InputLayer)            [(None, 128)]        0
__________________________________________________________________________________
tf_roberta_model (TFRobertaMode TFBaseModelOutputWit 124645632   input_3[0][0]
                                                                 input_4[0][0]
__________________________________________________________________________________
dense_1 (Dense)                 (None, 3)            2307        tf_roberta_model[0]
[1]
==================================================================================
Total params: 124,647,939
Trainable params: 124,647,939
Non-trainable params: 0
__________________________________________________________________________________
```

Start fine tuning the RoBERTa Algorithms

In [87]:
```
history_2 = model.fit([train_input_ids,train_attention_masks], y_train, validation_dat
```

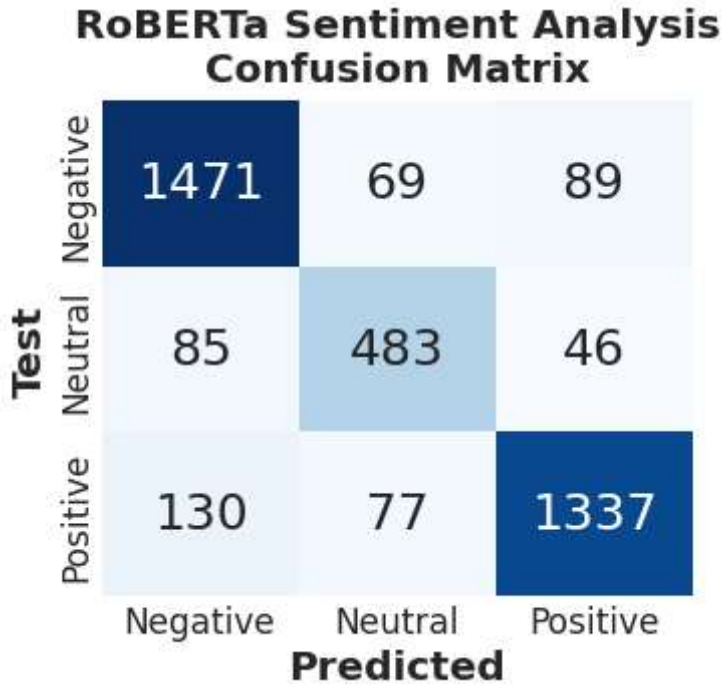
```
Epoch 1/4
1620/1620 [==============================] - 780s 473ms/step - loss: 0.5690 - categor
ical_accuracy: 0.7741 - val_loss: 0.5047 - val_categorical_accuracy: 0.8326
Epoch 2/4
1620/1620 [==============================] - 766s 473ms/step - loss: 0.3356 - categor
ical_accuracy: 0.8805 - val_loss: 0.3015 - val_categorical_accuracy: 0.8959
Epoch 3/4
1620/1620 [==============================] - 781s 482ms/step - loss: 0.2454 - categor
ical_accuracy: 0.9135 - val_loss: 0.2671 - val_categorical_accuracy: 0.9111
Epoch 4/4
1620/1620 [==============================] - 782s 482ms/step - loss: 0.1860 - categor
ical_accuracy: 0.9342 - val_loss: 0.2406 - val_categorical_accuracy: 0.9252
```

The results

In [88]:
```
result_roberta = model.predict([test_input_ids,test_attention_masks])
```

In [89]:
```
y_pred_roberta =  np.zeros_like(result_roberta)
y_pred_roberta[np.arange(len(y_pred_roberta)), result_roberta.argmax(1)] = 1
```

In [90]:
```
conf_matrix(y_test.argmax(1),y_pred_roberta.argmax(1),'RoBERTa Sentiment Analysis\nCor
```



In [91]:
```
print('\tClassification Report for RoBERTa:\n\n',classification_report(y_test,y_pred_r
```

```
        Classification Report for RoBERTa:

              precision    recall  f1-score   support

    Negative       0.87      0.90      0.89      1629
     Neutral       0.77      0.79      0.78       614
    Positive       0.91      0.87      0.89      1544

   micro avg       0.87      0.87      0.87      3787
   macro avg       0.85      0.85      0.85      3787
weighted avg       0.87      0.87      0.87      3787
 samples avg       0.87      0.87      0.87      3787
```

Results Summary

BERT Classification Report

In [92]:
```
print('Classification Report for BERT:\n',classification_report(y_test,y_pred_bert, ta
```

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

    Negative       0.87      0.92      0.90      1629
     Neutral       0.83      0.79      0.81       614
    Positive       0.92      0.89      0.90      1544

   micro avg       0.89      0.89      0.89      3787
   macro avg       0.87      0.86      0.87      3787
weighted avg       0.89      0.89      0.88      3787
 samples avg       0.89      0.89      0.89      3787
```

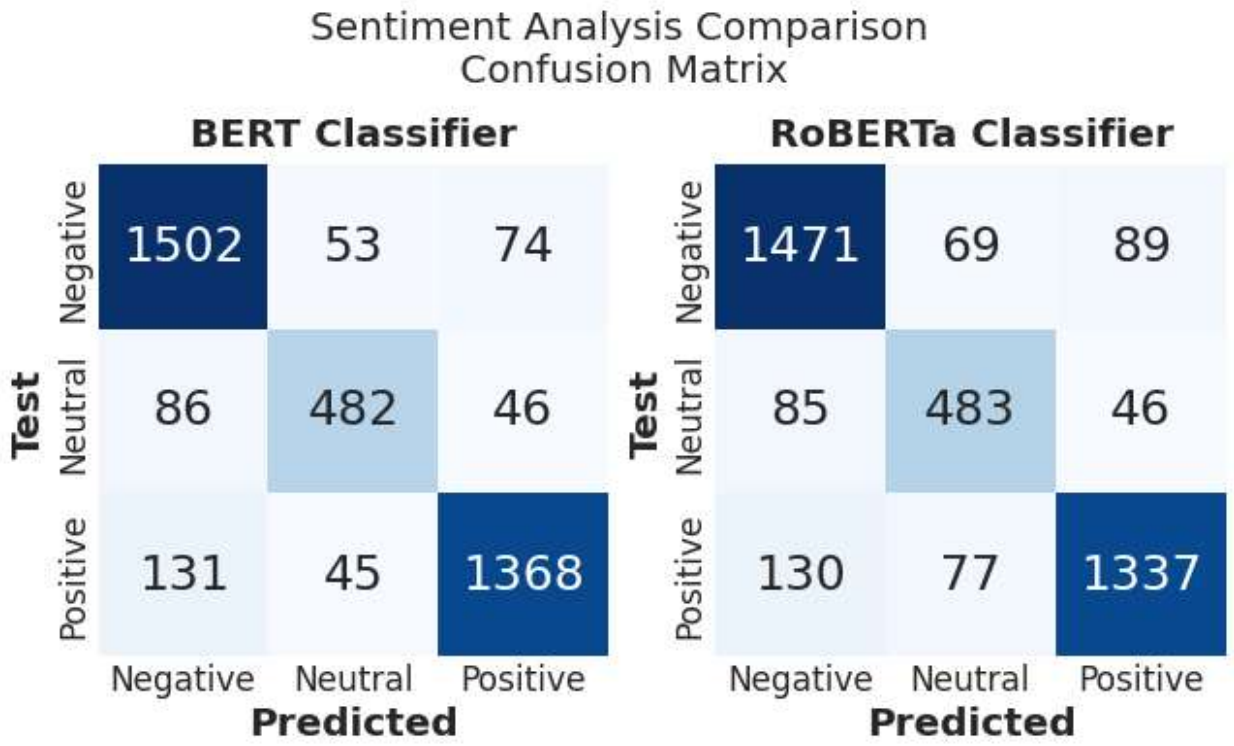
RoBERTa Classification Report

In [93]:
```
print('Classification Report for RoBERTa:\n',classification_report(y_test,y_pred_rober
```

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

    Negative       0.87      0.90      0.89      1629
     Neutral       0.77      0.79      0.78       614
    Positive       0.91      0.87      0.89      1544

   micro avg       0.87      0.87      0.87      3787
   macro avg       0.85      0.85      0.85      3787
weighted avg       0.87      0.87      0.87      3787
 samples avg       0.87      0.87      0.87      3787
```

Classification Matrix Comparison

In [94]:
```
fig, ax = plt.subplots(1,2,figsize=(9,5.5))

labels = ['Negative', 'Neutral', 'Positive']
plt.suptitle('Sentiment Analysis Comparison\n Confusion Matrix', fontsize=20)


sns.heatmap(confusion_matrix(y_test.argmax(1),y_pred_bert.argmax(1)), annot=True, cmap

ax[0].set_title('BERT Classifier', fontsize=20)
ax[0].set_yticklabels(labels, fontsize=17);
ax[0].set_xticklabels(labels, fontsize=17);
ax[0].set_ylabel('Test', fontsize=20)
ax[0].set_xlabel('Predicted', fontsize=20)

sns.heatmap(confusion_matrix(y_test.argmax(1),y_pred_roberta.argmax(1)), annot=True, c
ax[1].set_title('RoBERTa Classifier', fontsize=20)
ax[1].set_yticklabels(labels, fontsize=17);
ax[1].set_xticklabels(labels, fontsize=17);
ax[1].set_ylabel('Test', fontsize=20)
ax[1].set_xlabel('Predicted', fontsize=20)


plt.show()
```



We can observe that both algorithms performed admirably on the classification task.