YouTube Comments Sentiment Analysis

Import packages

In [1]:

```
import pandas as pd; import os
import csv; import numpy as np
import re; import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# training data
okgo = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\OKGO.
trump = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\trum
swift = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\Tayl
royal = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\Roya
paul = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\Logar
```

In [3]:

```
blogs = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\Kage
tweets = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_datasets\\twi
```

Data Preprocessing

In [4]:

```
# clean dataframes
tweets = tweets.drop(['Topic', 'TweetId', "TweetDate"], axis = 1).dropna()
tweets.head()
```

Out[4]:

	Sentiment	TweetText
0	positive	Now all @Apple has to do is get swype on the i...
1	positive	@Apple will be adding more carrier support to ...
2	positive	Hilarious @youtube video - guy does a duet wit...
3	positive	@RIM you made it too easy for me to switch to ...
4	positive	I just realized that the reason I got into twi...

In [5]:

```
def fix_cols(DF):
    DF = DF.iloc[:,:2]
    DF.columns = ["label", "comment"]
    return DF
```

In [6]:

```
okgo = fix_cols(okgo)
trump = fix_cols(trump)
swift = fix_cols(swift)
royal = fix_cols(royal)
paul = fix_cols(paul)
tweets = fix_cols(tweets)


okgo.head()
```

Out[6]:

	label	comment
0	-1.0	Everyone knows brand's papers from.\rBut -No o...
1	0.0	ÒYour paper cut balance is: \r-£252791027710Ó
2	1.0	OH SHIT WHEN I SAW THIS ON MY FRONT PAGE.........
3	1.0	Blowing my mind yet again
4	0.0	Should have gone with Dunder Mifflin

In [7]:
```
tweets.label = tweets.label.replace({'positive': '1.0', 'negative':'-1.0', 'neutral
tweets['label'] = pd.to_numeric(tweets['label'], errors='coerce')
```

In [8]:
```
tweets = fix_cols(tweets)
blogs = fix_cols(blogs)

tweets.head()
```

Out[8]:

	label	comment
0	1.0	Now all @Apple has to do is get swype on the i...
1	1.0	@Apple will be adding more carrier support to ...
2	1.0	Hilarious @youtube video - guy does a duet wit...
3	1.0	@RIM you made it too easy for me to switch to ...
4	1.0	I just realized that the reason I got into twi...

Create Datasets

In [9]:
```
yt_comments = pd.concat([okgo, trump, swift, royal, paul], ignore_index=True)
yt_comments.head()
```

Out[9]:

	label	comment
0	-1.0	Everyone knows brand's papers from.\rBut -No o...
1	0.0	ÒYour paper cut balance is: \r-£252791027710Ó
2	1.0	OH SHIT WHEN I SAW THIS ON MY FRONT PAGE.........
3	1.0	Blowing my mind yet again
4	0.0	Should have gone with Dunder Mifflin

In [10]:
```
non_yt_comments = pd.concat([blogs, tweets], ignore_index=True)
non_yt_comments.head()
```

Out[10]:

	label	comment
0	1.0	i liked the Da Vinci Code a lot
1	1.0	i liked the Da Vinci Code a lot
2	1.0	I liked the Da Vinci Code but it ultimatly di...
3	1.0	that's not even an exaggeration) and at midn...
4	1.0	I loved the Da Vinci Code but now I want some...

In [11]:

```
comments = pd.concat([yt_comments, non_yt_comments], ignore_index=True)
comments.head()
```

Out[11]:

	label	comment
0	-1.0	Everyone knows brand's papers from.\rBut -No o...
1	0.0	ÒYour paper cut balance is: \r-£25279102771Ó
2	1.0	OH SHIT WHEN I SAW THIS ON MY FRONT PAGE.........
3	1.0	Blowing my mind yet again
4	0.0	Should have gone with Dunder Mifflin

Remove Non-Alphabetic Characters (including numbers)

In [12]:

```
def convert_to_string(DF):
    DF["comment"]= DF["comment"].astype(str)
```

In [13]:

```
convert_to_string(comments)
```

In [15]:

```
def cleanerFn(b):
    # keeps only words with alphabetic characters in comments
    for row in range(len(b)):
        line = b.loc[row, "comment"]
        b.loc[row,"comment"] = re.sub("[^a-zA-Z]", " ", line)
```

In [16]:

```
cleanerFn(comments)
comments.head()
```

Out[16]:

	label	comment
0	-1.0	Everyone knows brand s papers from But No on...
1	0.0	Your paper cut balance is
2	1.0	OH SHIT WHEN I SAW THIS ON MY FRONT PAGE ...
3	1.0	Blowing my mind yet again
4	0.0	Should have gone with Dunder Mifflin

Natural Language Processing

In [17]:

```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import *
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [17]:

```
sw = stopwords.words('english')
ps = PorterStemmer()
lemmatizer = nltk.stem.WordNetLemmatizer()
```

Tokenization, Remove Stop Words, Lemmatization & Stemming

In [18]:
```
def nlpFunction(DF):
    DF['com_token'] = DF['comment'].str.lower().str.split()
    DF['com_remv'] = DF['com_token'].apply(lambda x: [y for y in x if y not in sw])
    DF["com_lemma"] = DF['com_remv'].apply(lambda x : [lemmatizer.lemmatize(y) for
    DF['com_stem'] = DF['com_lemma'].apply(lambda x : [ps.stem(y) for y in x]) # st
    DF["com_tok_str"] = DF["com_stem"].apply(', '.join)
    DF["com_full"] = DF["com_remv"].apply(' '.join)
    return DF
```

In [19]:
```
comments = nlpFunction(comments)
comments.head()
```

Out[19]:

	label	comment	com_token	com_remv	com_lemma	com_stem	com_tok_str	com_full
0	-1.0	Everyone knows brand s papers from But No on...	[everyone, knows, brand, s, papers, from, but,...	[everyone, knows, brand, papers, one, knows, w...	[everyone, know, brand, paper, one, know, welf...	[everyon, know, brand, paper, one, know, welfa...	everyon, know, brand, paper, one, know, welfar...	everyone knows brand papers one knows welfare ...
1	0.0	Your paper cut balance is	[your, paper, cut, balance, is]	[paper, cut, balance]	[paper, cut, balance]	[paper, cut, balanc]	paper, cut, balanc	paper cut balance
2	1.0	OH SHIT WHEN I SAW THIS ON MY FRONT PAGE ...	[oh, shit, when, i, saw, this, on, my, front, ...	[oh, shit, saw, front, page, love, song]	[oh, shit, saw, front, page, love, song]	[oh, shit, saw, front, page, love, song]	oh, shit, saw, front, page, love, song	oh shit saw front page love song
3	1.0	Blowing my mind yet again	[blowing, my, mind, yet, again]	[blowing, mind, yet]	[blowing, mind, yet]	[blow, mind, yet]	blow, mind, yet	blowing mind yet
4	0.0	Should have gone with Dunder Mifflin	[should, have, gone, with, dunder, mifflin]	[gone, dunder, mifflin]	[gone, dunder, mifflin]	[gone, dunder, mifflin]	gone, dunder, mifflin	gone dunder mifflin

In [20]:
```
def drop_cols_after_nlp(comments):
    comments = comments.drop(columns = ['comment', 'com_token', 'com_remv', 'com_le
    return comments
comments = drop_cols_after_nlp(comments)
comments.head()
```

Out[20]:

	label	com_full
0	-1.0	everyone knows brand papers one knows welfare ...
1	0.0	paper cut balance
2	1.0	oh shit saw front page love song
3	1.0	blowing mind yet
4	0.0	gone dunder mifflin

In [21]:
```
comments.rename(columns = {'com_full': 'comment'}, inplace=True)
comments.head()
```

Out[21]:

	label	comment
0	-1.0	everyone knows brand papers one knows welfare ...
1	0.0	paper cut balance
2	1.0	oh shit saw front page love song
3	1.0	blowing mind yet
4	0.0	gone dunder mifflin

In [22]:

```python
def remove_missing_vals(comments):
    comments['comment'] = comments['comment'].str.strip()
    comments = comments[comments.comment != 'nan'] # remove nan values from data
    comments = comments[comments.comment != '']

remove_missing_vals(comments)
```

In [23]:

```python
comments.head()
```

Out[23]:

	label	comment
0	-1.0	everyone knows brand papers one knows welfare ...
1	0.0	paper cut balance
2	1.0	oh shit saw front page love song
3	1.0	blowing mind yet
4	0.0	gone dunder mifflin

In [24]:

```python
comments['label'].isna().sum()
```

Out[24]:

```
2355
```

In [25]:

```python
comments = comments[comments['label'].notna()]
comments['label'].isna().sum()
```

Out[25]:

```
0
```

In [26]:

```python
len(comments)
```

Out[26]:

```
14830
```

In [27]:

```python
X = comments['comment']
y = comments.label
```

In [28]:

```python
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=53, test_siz
```

Vectorize the tweets

We have the training and testing data all set up, but we need to create vectorized representations of the tweets in order to apply machine learning.

To do so, we will utilize the `CountVectorizer` and `TfidfVectorizer` classes which we will first need to fit to the data.

Once this is complete, we can start modeling with the new vectorized tweets!

In [29]:

```python
# Initialize count vectorizer
count_vectorizer = CountVectorizer(stop_words='english',
                                   min_df=0.05, max_df=0.9)

# Create count train and test variables
count_train = count_vectorizer.fit_transform(X_train)
count_test = count_vectorizer.transform(X_test)

# Initialize tfidf vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english',
                                   min_df=0.05, max_df=0.9)

# Create tfidf train and test variables
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)
```

Model Building

In [30]:

```python
# Set seed for reproducibility
import random; random.seed(5)

# Import all we need from sklearn
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn import metrics
```

Multinomial Naive-Bayes Model

Training a multinomial naive Bayes model

Now that we have the data in vectorized form, we can train the first model. Investigate using the Multinomial Naive Bayes model with both the `CountVectorizer` and `TfidfVectorizer` data.

To assess the accuracies, we will print the test sets accuracy scores for both models.

In [31]:

```python
# Create a MulitnomialNB model
tfidf_nb = MultinomialNB()
tfidf_nb.fit(tfidf_train,y_train)
# Run predict on your TF-IDF test data to get your predictions
tfidf_nb_pred = tfidf_nb.predict(tfidf_test)

# Calculate the accuracy of your predictions
tfidf_nb_score = metrics.accuracy_score(y_test,tfidf_nb_pred)

# Create a MulitnomialNB model
count_nb = MultinomialNB()
count_nb.fit(count_train,y_train)

# Run predict on your count test data to get your predictions
count_nb_pred = count_nb.predict(count_test)
```

```
# Calculate the accuracy of your predictions
count_nb_score = metrics.accuracy_score(count_nb_pred,y_test)

print('NaiveBayes Tfidf Score: ', tfidf_nb_score)
print('NaiveBayes Count Score: ', count_nb_score)
```

```
NaiveBayes Tfidf Score:  0.7909924487594391
NaiveBayes Count Score:  0.7831715210355987
```

Logistic Regression

In [32]:

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(tfidf_train,y_train)
accuracy_lr = lr_model.score(tfidf_test,y_test)
print("Logistic Regression accuracy is (for Tfidf) :",accuracy_lr)
```

```
Logistic Regression accuracy is (for Tfidf) : 0.7880258899676376
```

In [33]:

```
lr_model = LogisticRegression()
lr_model.fit(count_train,y_train)
accuracy_lr = lr_model.score(count_test,y_test)
print("Logistic Regression accuracy is (for Count) :",accuracy_lr)
```

```
Logistic Regression accuracy is (for Count) : 0.7877562028047465
```

SVC

In [34]:

```
# Create a SVM model
from sklearn import svm
tfidf_svc = svm.SVC(kernel='linear', C=1)

tfidf_svc.fit(tfidf_train,y_train)
# Run predict on your tfidf test data to get your predictions
tfidf_svc_pred = tfidf_svc.predict(tfidf_test)

# Calculate your accuracy using the metrics module
tfidf_svc_score = metrics.accuracy_score(y_test,tfidf_svc_pred)

print("LinearSVC Score (for tfidf):   %0.3f" % tfidf_svc_score)
```

```
LinearSVC Score (for tfidf):   0.792
```

In [35]:

```
count_svc = svm.SVC(kernel='linear', C=1)

count_svc.fit(count_train,y_train)
# Run predict on your count test data to get your predictions
count_svc_pred = count_svc.predict(count_test)

# Calculate your accuracy using the metrics module
count_svc_score = metrics.accuracy_score(y_test,count_svc_pred)

print("LinearSVC Score (for Count):   %0.3f" % tfidf_svc_score)
```

```
LinearSVC Score (for Count):   0.792
```

Desicion Tree

In [36]:

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(tfidf_train,y_train)
```

```
accuracy_dt = dt_model.score(tfidf_test,y_test)
print("Decision Tree accuracy is (for Tfidf):",accuracy_dt)
```

```
Decision Tree accuracy is (for Tfidf): 0.7980043149946062
```

In [37]:

```
dt_model = DecisionTreeClassifier()
dt_model.fit(count_train,y_train)
accuracy_dt = dt_model.score(count_test,y_test)
print("Decision Tree accuracy is (for Count):",accuracy_dt)
```

```
Decision Tree accuracy is (for Count): 0.7977346278317152
```

Random Forest

In [38]:

```
from sklearn.ensemble import RandomForestClassifier
rf_model_initial = RandomForestClassifier(n_estimators = 5, random_state = 1)
rf_model_initial.fit(tfidf_train,y_train)
print("Random Forest accuracy for 5 trees is (Tfidf):",rf_model_initial.score(tfidf
```

```
Random Forest accuracy for 5 trees is (Tfidf): 0.7977346278317152
```

In [39]:

```
rf_model_initial = RandomForestClassifier(n_estimators = 5, random_state = 1)
rf_model_initial.fit(count_train,y_train)
print("Random Forest accuracy for 5 trees is (Count):",rf_model_initial.score(count
```

```
Random Forest accuracy for 5 trees is (Count): 0.7974649406688241
```

Predicting Sentiment For YouTube video

Reading Testing YouTube Video Comments

Comments.csv files has comments of youtube video

In [40]:

```
prediction_comments = pd.read_csv('C:\\Users\\NANDISH KUMAR\\OneDrive\\Desktop\\YT_
prediction_comments = prediction_comments.iloc[:,:1]
prediction_comments.columns=['comment']
prediction_comments.head()
```

Out[40]:

	comment
0	What do YOU think to the current state of Fold...
1	Well, finally someone who can compete with Sam...
2	I wanna see them attempt something like the Z-...
3	4:57 "And then actually coming with the charge...
4	Personally, for me this was one of, if not the...

In [41]:

```
# Lets use SVC to predict on our youtube video comments
prediction_comments.head()
```

Out[41]:

	comment
0	What do YOU think to the current state of Fold...
1	Well, finally someone who can compete with Sam...
2	I wanna see them attempt something like the Z-...
3	4:57 "And then actually coming with the charge...
4	Personally, for me this was one of, if not the...

In [42]:

```
len(prediction_comments['comment'])
```

Out[42]:

```
1001
```

In [43]:

```
convert_to_string(prediction_comments)
cleanerFn(prediction_comments)
prediction_comments = nlpFunction(prediction_comments)
prediction_comments = drop_cols_after_nlp(prediction_comments)
prediction_comments.rename(columns = {'com_full': 'comment'}, inplace=True)
remove_missing_vals(prediction_comments)
prediction_comments.head()
```

Out[43]:

	comment
0	think current state foldable phones check tesl...
1	well finally someone compete samsung market co...
2	wanna see attempt something like z flip someth...
3	actually coming charger respect xiaomi getting...
4	personally one best video ever made simple alw...

In [44]:

```
tfidf_pred = tfidf_vectorizer.transform(prediction_comments['comment'])
tfidf_svc_pred = tfidf_svc.predict(tfidf_pred)
```

In [45]:

```
neutral = (tfidf_svc_pred == 0.0).sum()
positive = (tfidf_svc_pred == 1.0).sum()
negative = (tfidf_svc_pred < 0).sum()
```

In [46]:

```
print(neutral, positive, negative)
```

```
833 161 7
```

In [47]:

```
print("Good video" if positive > negative else "Bad video")
```

```
Good video
```