# coding: utf-8

# In[ ]:

#for Atul

import tweepy

consumer\_key = "phDphxJphqdy1z6p4Ro9nXOws"

consumer\_secret = "AA9tzpkDxh6zLsgyz5bSHbLi5412uFYhct26cBBAFXsauPAXQP"

access\_token = "1077797923272810496-zdTr9xhZd1j7pOFEtL2wMQZhfWSmyr"

access\_token\_secret = "Jrm6xp6iEi1VSUD9cd6LXQU34lGu8N7v5aPVpNfjUqzim"

# In[1]:

#for sandeep

consumer\_key = "X9OBSOJTdVFm7JmLXmRXnELMA"

consumer\_secret = "kP2AtlyPTP3B5sBMrZU0d3PcDY681yvH6TcnJUIoh2xRM742ZU"

access\_token = "3149540395-Ul3fM3MPS8aaKnUCWWpMDZOkZ0nXxxl5fK17eMT"

access\_token\_secret = "PBwRg4QzVjnjYxo9u8BedNCyNTDEYEZrWiAoDy4S3SiSG"

# In[2]:

# Creating the authentication object

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

# Setting your access token and secret

auth.set\_access\_token(access\_token, access\_token\_secret)

# Creating the API object while passing in auth information

api = tweepy.API(auth)

# In[4]:

#ByKeyword

# Creating the API object while passing in auth information

api = tweepy.API(auth)

# The search term you want to find

query = "iphonex"

# Language code (follows ISO 639-1 standards)

language = "en"

# Calling the user\_timeline function with our parameters

results = api.search(q=query, lang=language,count=2000)

print(results)

# In[5]:

# foreach through all tweets pulled

for tweet in results:

# printing the text stored inside the tweet object

print (tweet.user.screen\_name,"Tweeted:",tweet.text,tweet.created\_at)

# Facing lot of issues in pulling data from twitter. earlier they did not provide us keys and then once we have provided isb id we got the keys. so we have decided to go ahead with kaggle data first. we will also use already pulled twitter data along with this kaggle data.

# Data from kaggle https://www.kaggle.com/kewalkishang/iphonexreview

# Data understanding

# In[13]:

comments\_list = pd.read\_csv(r'C:\Users\sasing\OneDrive - Adobe Systems Incorporated\ISB\CBA\Term1\Practicum\IPhoneReview.csv',encoding='latin-1')

# In[14]:

len(comments\_list)

# In[15]:

# Three sample comments

comments\_list[:3]

# Data summary and understanding to create a model on review data so we could find a better perspective on our test data pulled from twitter and kaggle

# Here is the subset of unlocked phone data from Amazon. Original kaggle data contains more than 400K rows so we have pulled only small subset which seems to be more relevant for our analysis

# In[35]:

#Data Prep for Sentiment Analysis

import pandas as pd

import numpy as np

# Read in the data

df1 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/Amazon\_Unlocked\_Mobile\_Reviews1.csv')

df2 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/Amazon\_Unlocked\_Mobile\_Reviews2.csv')

df=pd.concat([df1,df2],ignore\_index=True)

df.head()

# In[36]:

# Drop missing values

df.dropna(inplace=True)

# Remove any 'neutral' ratings equal to 3

df = df[df['Rating'] != 3]

# Encode 4s and 5s as 1 (rated positively)

# Encode 1s and 2s as 0 (rated poorly)

df['Positively Rated'] = np.where(df['Rating'] > 3, 1, 0)

df.head()

# In[37]:

from sklearn.model\_selection import train\_test\_split

# Split data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['Reviews'],

df['Positively Rated'],random\_state=3)

# In[38]:

#CountVectorizer, n-grams

from sklearn.feature\_extraction.text import CountVectorizer

vect = CountVectorizer(min\_df=5, ngram\_range=(1,2)).fit(X\_train)

X\_train\_vectorized = vect.transform(X\_train)

len(vect.get\_feature\_names())

# In[39]:

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import roc\_auc\_score

model = LogisticRegression()

model.fit(X\_train\_vectorized, y\_train)

predictions = model.predict(vect.transform(X\_test))

print('ROC\_AUC: ', roc\_auc\_score(y\_test, predictions))

# In[40]:

feature\_names = np.array(vect.get\_feature\_names())

sorted\_coef\_index = model.coef\_[0].argsort()

print('Smallest Coefs:\n{}\n'.format(feature\_names[sorted\_coef\_index[:10]]))

print('Largest Coefs: \n{}'.format(feature\_names[sorted\_coef\_index[:-11:-1]]))

# In[41]:

# check the model to see if it correctly identifies a positive review

print(model.predict(vect.transform(['not an issue, phone is working',

'an issue, phone is not working'])))

# In[42]:

#Sentiment Analysis for iPhone X

comments\_positivity = model.predict(vect.transform(comments\_list))

print(comments\_positivity)

# In[43]:

comments\_positivity.mean()

# In[44]:

from wordcloud import WordCloud, STOPWORDS

import matplotlib.pyplot as plt

import pandas as pd

df = comments\_list

# In[45]:

comment\_words = ' '

stopwords = set(STOPWORDS)

# iterate through the csv file

for val in df.Review:

# typecaste each val to string

val = str(val)

# split the value

tokens = val.split()

# Converts each token into lowercase

for i in range(len(tokens)):

tokens[i] = tokens[i].lower()

for words in tokens:

comment\_words = comment\_words + words + ' '

# In[46]:

wordcloud = WordCloud(width = 800, height = 800,

background\_color ='white',

stopwords = stopwords,

min\_font\_size = 10).generate(comment\_words)

# plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor = None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.show()

# In[48]:

df.head()

# In[49]:

from textblob import TextBlob

from nltk.tokenize import sent\_tokenize

# In[50]:

review = df.Review.astype(str)

review.head()

# In[72]:

def sentiment\_calc(text):

try:

return TextBlob(text).sentiment[0]

except:

return None

df['sentiment'] = df['Review'].apply(sentiment\_calc)

# In[73]:

df.head()

# In[68]:

dfs = df['Review'].to\_string()

# In[71]:

dfs

# In[74]:

dfm = df.sentiment.mean()

# In[75]:

dfm

# In[113]:

def sentiment\_type(i):

try:

if i > 0:

return 'positive'

elif i < 0:

return 'negative'

else:

return 'neutral'

except:

return None

df['sentiment1'] = df['sentiment'].apply(sentiment\_type)

# In[114]:

df.head()

# In[118]:

df['sentiment'].plot(kind='bar')

# In[119]:

df['sentiment1'].value\_counts().plot(kind='bar')

# In[120]:

from statsmodels.graphics.mosaicplot import mosaic

plt.rcParams['font.size'] = 16.0

mosaic(df, ['sentiment1']);

# Pulling the twitter data for analysis. till now we have used review data from Kaggle but as mentioned above we have also pulled some tweets since December. so we will pull all files, merge them and try to find the sentiment about iphonex

# In[123]:

df0 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

df1 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result1.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

df2 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result2.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

df3 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result3.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

df4 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result4.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

df5 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result5.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

#df6 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result6.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

df7 = pd.read\_csv('https://raw.githubusercontent.com/siwanian/PracticumData/master/result7.csv',names=['created','screenname','followers','favourites','text'], error\_bad\_lines=False)

# In[124]:

dff = pd.concat([df0,df1,df2,df3,df4,df5,df7],ignore\_index=True)

# In[125]:

dff.head()

# In[126]:

#Sentiment Analysis for iPhone X through model built above

twitter\_positive = model.predict(vect.transform(dff))

print(twitter\_positive)

# In[127]:

comments\_positivity.mean()

# Tokenizing the words so we could see the frequency of everthing along with iphone x in a wordcloud

# In[128]:

tweet\_words = ' '

stopwords = set(STOPWORDS)

# iterate through the csv file

for val in dff.text:

# typecaste each val to string

val = str(val)

# split the value

tokens = val.split()

# Converts each token into lowercase

for i in range(len(tokens)):

tokens[i] = tokens[i].lower()

for words in tokens:

tweet\_words = tweet\_words + words + ' '

# Creating wordcloud

# In[129]:

wordcloud = WordCloud(width = 800, height = 800,

background\_color ='white',

stopwords = stopwords,

min\_font\_size = 10).generate(tweet\_words)

# plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor = None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.show()

# We can see through this word cloud that most of the space is covered by html tags and other things. we have not splitted the model data to exclude most frequent words so we do not lose some important information

# In[141]:

from sklearn.decomposition import LatentDirichletAllocation,PCA

from sklearn import svm

from sklearn.linear\_model import LogisticRegression

from sklearn.feature\_extraction.text import CountVectorizer,TfidfVectorizer

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import train\_test\_split

from nltk.tokenize import sent\_tokenize, word\_tokenize

from nltk.stem import WordNetLemmatizer

from nltk import pos\_tag

# Boiler place code for cleanup. Run this on twitter data and find how word cloud looks now

# In[142]:

def cleanText(text):

text = re.sub('[^ a-zA-Z]','',text)

text = re.sub(r' +', ' ', text)

return text

wnl = WordNetLemmatizer()

def lemmatizeSentences(data):

count = 0

for text in data:

# clean the text

text = cleanText(text)

sentence = ""

for i, j in pos\_tag(word\_tokenize(text)):

word = ''

if j[0].lower() in ['n', 'v', 'r']:

word = wnl.lemmatize(i, j[0].lower())

elif j[0].lower() is 'j':

word = wnl.lemmatize(i, 'a')

else:

word = wnl.lemmatize(i)

sentence = sentence + " " + word.lower()

data[count] = sentence

count = count + 1;

return data

# In[144]:

dff['clean'] = lemmatizeSentences(dff['text'])

dff['clean'].head()

# In[145]:

tweet\_words = ' '

stopwords = set(STOPWORDS)

# iterate through the csv file

for val in dff.clean:

# typecaste each val to string

val = str(val)

# split the value

tokens = val.split()

# Converts each token into lowercase

for i in range(len(tokens)):

tokens[i] = tokens[i].lower()

for words in tokens:

tweet\_words = tweet\_words + words + ' '

# In[146]:

wordcloud = WordCloud(width = 800, height = 800,

background\_color ='white',

stopwords = stopwords,

min\_font\_size = 10).generate(tweet\_words)

# plot the WordCloud image

plt.figure(figsize = (8, 8), facecolor = None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.show()

# After cleanup we can see that wordcloud provides us much more information than previous cloud. we will go with this data and use the sentiment

# In[147]:

def sentiment\_calc(text):

try:

return TextBlob(text).sentiment[0]

except:

return None

dff['sentiment'] = dff['clean'].apply(sentiment\_calc)

# In[148]:

dff.head()

# In[149]:

def sentiment\_type(i):

try:

if i > 0:

return 'positive'

elif i < 0:

return 'negative'

else:

return 'neutral'

except:

return None

df['sentiment1'] = df['sentiment'].apply(sentiment\_type)

# In[150]:

df['sentiment'].plot(kind='bar')

# In[151]:

df['sentiment1'].value\_counts().plot(kind='bar')

# In[152]:

from statsmodels.graphics.mosaicplot import mosaic

plt.rcParams['font.size'] = 16.0

mosaic(df, ['sentiment1']);