

|  |  |
| --- | --- |
| **Name: Sayyed Sohail Rashid** | **Course Name: SMA-LAB** |
| **Class: BE-CO** | **Batch: 01** |
| **Roll no: 18CO48** | **Experiment No: 05** |

**Aim**: Develop Content (text, emoticons, image, audio, video) based social media analytics model for business.

**Code:**

# -\*- coding: utf-8 -\*-

!pip install snscrape

# Commented out IPython magic to ensure Python compatibility.

import pandas as pd

import snscrape.modules.twitter as sntwitter

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import WordNetLemmatizer

from nltk.stem.porter import PorterStemmer

import string

import re

import textblob

from textblob import TextBlob

import os

from wordcloud import WordCloud, STOPWORDS

from wordcloud import ImageColorGenerator

import warnings

# %matplotlib inline

os.system("snscrape --jsonl --max-results 5000 --since 2023-01-31 twitter-search 'Unilever 2023 until:2023-02-07'>text-query-tweets.json")

tweets\_df = pd.read\_json("text-query-tweets.json" ,lines=True)

tweets\_df.tail(5)

tweets\_df.to\_csv()

df1 = tweets\_df[[ 'date', 'rawContent' , 'renderedContent' , 'user' , 'replyCount' ,'retweetCount' , 'likeCount' , 'lang' , 'place']]

df1.head()

plt.figure(figsize=(17, 5))

sns.heatmap(df1.isnull(), cbar=True, yticklabels=False)

plt.xlabel("Column\_Name", size=14, weight="bold")

plt.title("Places of missing values in column",size=17)

plt.show()

import plotly.graph\_objects as go

Top\_Location\_Of\_tweet= df1['place'].value\_counts().head (10)

print(Top\_Location\_Of\_tweet)

from nltk. corpus import stopwords

stop = stopwords.words('english')

df1['renderedContent'].apply(lambda x: [item for item in x if item not in stop])

df1.shape

!pip install tweet-preprocessor

#Remove unnecessary characters

punct = ['%','/',':','\\','&amp','&',';','?']

def remove\_punctuations(text):

  for punctuation in punct:

    text = text.replace(punctuation,'')

  return text

df1['renderedContent'] = df1['renderedContent'].apply(lambda x: remove\_punctuations(x))

df1['renderedContent'].replace( '', np.nan, inplace=True)

df1.dropna(subset=["renderedContent"],inplace=True)

len(df1)

df1 = df1.reset\_index(drop=True)

df1.head()

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

# Commented out IPython magic to ensure Python compatibility.

sns.set\_style('whitegrid')

# %matplotlib inline

stop=stop+['assured' , 'health' , 'http' , '2023', 'best' ,'look' , 'union', 'product' , 'customer' , 'india']

def plot\_20\_most\_common\_words(count\_data, count\_vectorizer) :

  import matplotlib. pyplot as plt

  words = count\_vectorizer.get\_feature\_names()

  total\_counts = np. zeros(len(words))

  for t in count\_data:

    total\_counts = t.toarray()[0]

  count\_dict = (zip(words, total\_counts))

  count\_dict = sorted(count\_dict, key=lambda x:x[1],reverse=True)[0:20]

  words = [w[0] for w in count\_dict]

  counts = [w[1] for w in count\_dict]

  x\_pos = np.arange(len(words))

  plt.figure(2, (40,40))

  plt.subplot(title = '20 most common words')

  sns. set\_context('notebook',font\_scale=4,rc={ 'lines.linewidth' :2.5})

  sns.barplot(x\_pos, counts, palette='husl')

  plt.xticks(x\_pos, words, rotation=90)

  plt.xlabel('words')

  plt.ylabel('counts')

  plt.show()

count\_vectorizer = CountVectorizer(stop\_words=stop)

# Fit and transform the processed titles

count\_data = count\_vectorizer.fit\_transform(df1['renderedContent'])

# print(count\_vectorizer)

# print(count\_data)

# Visualise the 20 most common words

plot\_20\_most\_common\_words(count\_data,count\_vectorizer)

plt.savefig( 'saved\_figure.png')

import cufflinks as cf

cf.go\_offline()

cf.set\_config\_file(offline=False, world\_readable=True)

def get\_top\_n\_bigram(corpus, n=None) :

  vec = CountVectorizer(ngram\_range=(2, 4), stop\_words="english").fit(corpus)

  bag\_of\_words = vec.transform(corpus)

  sum\_words = bag\_of\_words.sum(axis=0)

  words\_freq =[(word, sum\_words[0, idx]) for word, idx in vec.vocabulary\_.items()]

  words\_freq =sorted(words\_freq, key = lambda x: x[1], reverse=True)

  return words\_freq[:n]

common\_words = get\_top\_n\_bigram(df1['renderedContent'] , 8)

mydict={}

for word, freq in common\_words:

  bigram\_df = pd.DataFrame(common\_words,columns = ['ngram', 'count'])

bigram\_df.groupby( 'ngram' ).sum()['count'].sort\_values(ascending=False).sort\_values().plot.barh(title = 'Top 8 bigrams',color='orange')

def get\_subjectivity(text):

  return TextBlob(text).sentiment.subjectivity

def get\_polarity(text):

  return TextBlob(text).sentiment.polarity

df1['subjectivity']=df1[ 'renderedContent'].apply(get\_subjectivity)

df1[ 'polarity' ]=df1[ 'renderedContent'].apply(get\_polarity)

df1.head()

df1['textblob\_score'] =df1[ 'renderedContent'].apply(lambda x: TextBlob(x).sentiment.polarity)

neutral\_threshold=0.05

df1['textblob\_sentiment']=df1[ 'textblob\_score']

textblob\_df = df1[['renderedContent','textblob\_sentiment','likeCount']]

textblob\_df["textblob\_sentiment"].value\_counts()

df\_positive=textblob\_df[textblob\_df['textblob\_sentiment']=='positive' ]

df\_very\_positive=df\_positive[df\_positive['likeCount']>0]

df\_negative=textblob\_df[textblob\_df['textblob\_sentiment']=='Negative' ]

df\_neutral=textblob\_df[textblob\_df['textblob\_sentiment']=='Neutral' ]

#Creating the text variable

positive\_tw ="Social,Media,Analytics,Calculation,productivity,financially,physically,emotionally,socially,privacy,Brief,Method,Texts,Analysis"

# Creating word \_ cloud with text as argument in . generate() rtpthod

word\_cloud1 = WordCloud(collocations = False, background\_color = 'white') .generate(positive\_tw)

# Display the generated Word Cloud

plt. imshow(word\_cloud1, interpolation='bilinear')

plt.axis('off')

plt.show()

#Creating the text variable

negative\_tw ="productivity,financially,physically,emotionally,socially,privacy,neutrality,algorithms ,Social,Media,Analytics,Calculation, "

# Creating word \_ cloud with text as argument in . generate() rtpthod

word\_cloud2 = WordCloud(collocations = False, background\_color = 'white') .generate(negative\_tw)

# Display the generated Word Cloud

plt. imshow(word\_cloud2, interpolation='bilinear')

plt.axis('off')

plt.show()

#Creating the text variable

neutral\_tw ="neutrality,algorithms ,Social,Media,Analytics,Calculation,productivity,financially,physically"

# Creating word \_ cloud with text as argument in . generate() rtpthod

word\_cloud2 = WordCloud(collocations = False, background\_color = 'white') .generate(neutral\_tw)

# Display the generated Word Cloud

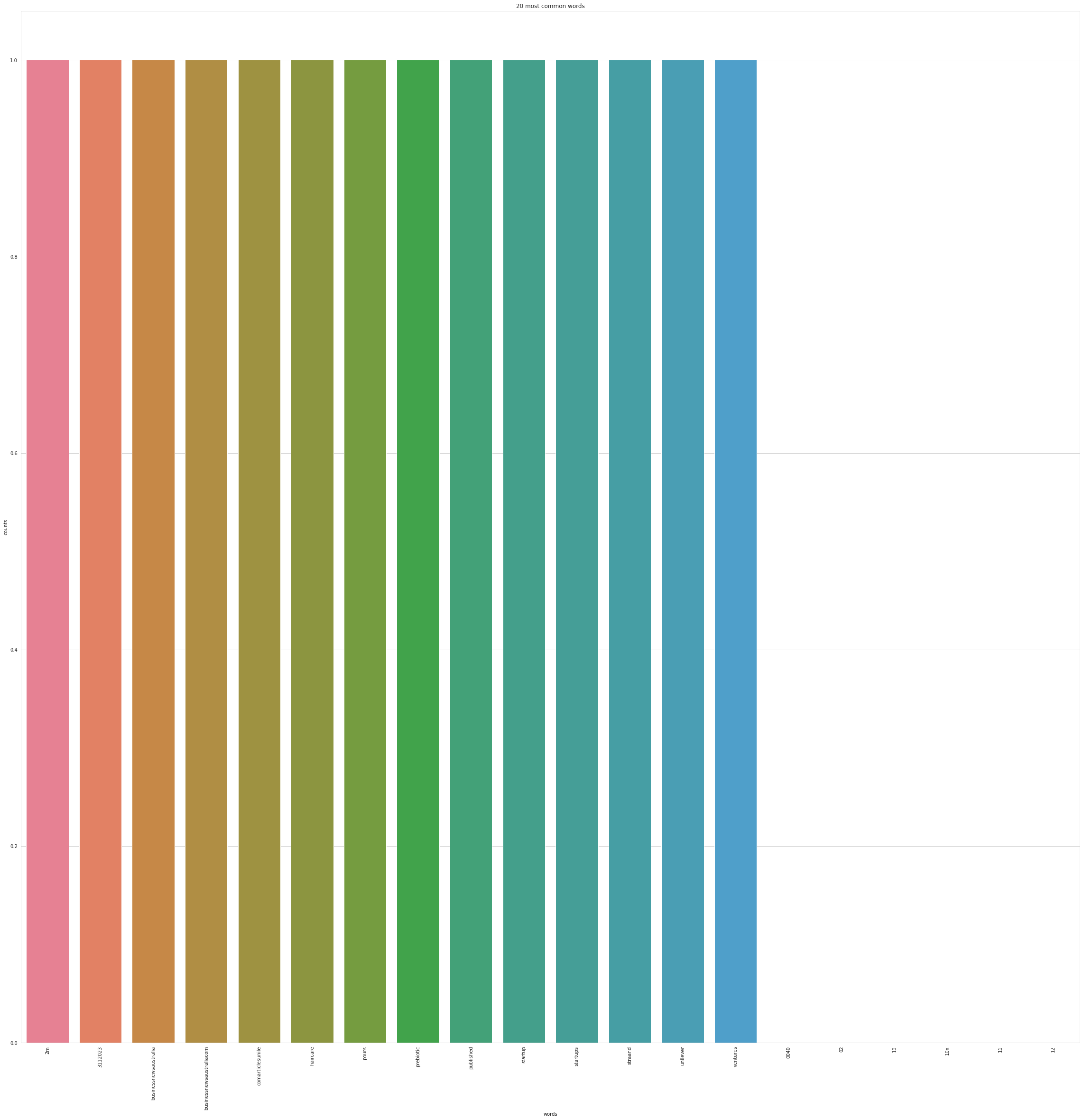
plt. imshow(word\_cloud2, interpolation='bilinear')

plt.axis('off')

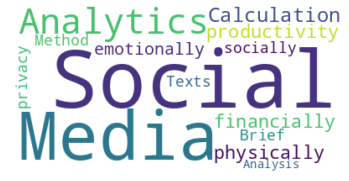
plt.show()

**Output:**

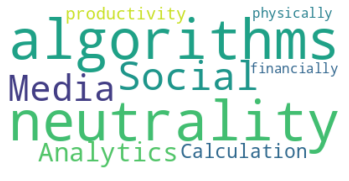












**Conclusion:**  We have successfully created content based social media analytics model for business.