#Sentiment anaysis on "Noise colourfit watch" reviews in amazon

import requests # Importing requests to extract content from a url

from bs4 import BeautifulSoup as bs # Beautifulsoup is for web scrapping...used to scrap specific content

import re

#pip install wordcloud

import matplotlib.pyplot as plt

from wordcloud import WordCloud

# creating empty reviews list

noise\_reviews=[]

for i in range(1,21):

ip=[]

url="https://www.amazon.in/Noise-ColorFit-Monitor-Personalised-Waterproof/product-reviews/B08PZ96F8V/ref=cm\_cr\_getr\_d\_paging\_btm\_next\_2?ie=UTF8&reviewerType=all\_reviews&pageNumber="+str(i)

response = requests.get(url)

soup = bs(response.content,"html.parser")# creating soup object to iterate over the extracted content

reviews = soup.find\_all("span",attrs={"class","a-size-base review-text review-text-content"})# Extracting the content under specific tags

for i in range(len(reviews)):

ip.append(reviews[i].text)

noise\_reviews=noise\_reviews+ip # adding the reviews of one page to empty list which in future contains all the reviews

# writng reviews in a text file

with open("noise\_py.txt","w",encoding='utf8') as output:

output.write(str(noise\_reviews))

# Joinining all the reviews into single paragraph

ip\_rev\_string = " ".join(noise\_reviews)

import nltk

#from nltk.corpus import stopwords

# Removing unwanted symbols incase if exists

ip\_rev\_string = re.sub("[^A-Za-z" "]+"," ", ip\_rev\_string).lower()

ip\_rev\_string = re.sub("[0-9" "]+"," ", ip\_rev\_string)

# words that contained in iphone XR reviews

ip\_reviews\_words = ip\_rev\_string.split(" ")

#TFIDF

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(ip\_reviews\_words, use\_idf=True,ngram\_range=(1, 3))

X = vectorizer.fit\_transform(ip\_reviews\_words)

with open("C:/Users/USER/Desktop/Data Mining/Text\_Mining/stop.txt","r") as sw:

stop\_words = sw.read()

stop\_words = stop\_words.split("\n")

stop\_words.extend(["noise","watch","product","good","app"])

ip\_reviews\_words = [w for w in ip\_reviews\_words if not w in stop\_words]

# Joinining all the reviews into single paragraph

ip\_rev\_string = " ".join(ip\_reviews\_words)

# WordCloud can be performed on the string inputs.

# Corpus level word cloud

wordcloud\_ip = WordCloud(

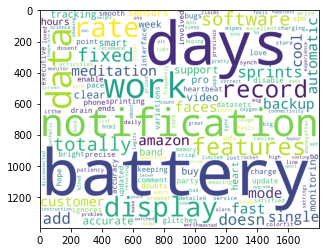
background\_color='White',

width=1800,

height=1400

).generate(ip\_rev\_string)

plt.imshow(wordcloud\_ip)



# positive words # Choose the path for +ve words stored in system

with open("C:/Users/USER/Desktop/Data Mining/Text\_Mining/positive-words.txt","r") as pos:

poswords = pos.read().split("\n")

# Positive word cloud

# Choosing the only words which are present in positive words

ip\_pos\_in\_pos = " ".join ([w for w in ip\_reviews\_words if w in poswords])

wordcloud\_pos\_in\_pos = WordCloud(

background\_color='White',

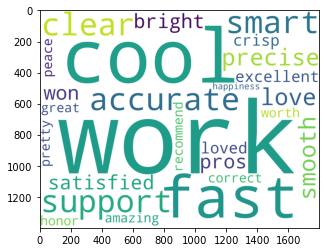
width=1800,

height=1400

).generate(ip\_pos\_in\_pos)

plt.figure(2)

plt.imshow(wordcloud\_pos\_in\_pos)



# negative words Choose path for -ve words stored in system

with open("C:/Users/USER/Desktop/Data Mining/Text\_Mining/negative-words.txt", "r") as neg:

negwords = neg.read().split("\n")

# negative word cloud

# Choosing the only words which are present in negwords

ip\_neg\_in\_neg = " ".join ([w for w in ip\_reviews\_words if w in negwords])

wordcloud\_neg\_in\_neg = WordCloud(

background\_color='black',

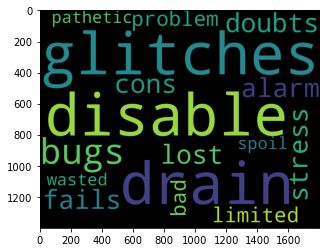
width=1800,

height=1400

).generate(ip\_neg\_in\_neg)

plt.figure(3)

plt.imshow(wordcloud\_neg\_in\_neg)



# wordcloud with bigram

nltk.download('punkt')

from wordcloud import WordCloud, STOPWORDS

WNL = nltk.WordNetLemmatizer()

# Lowercase and tokenize

text = ip\_rev\_string.lower()

# Remove single quote early since it causes problems with the tokenizer.

text = text.replace("'", "")

tokens = nltk.word\_tokenize(text)

text1 = nltk.Text(tokens)

# Remove extra chars and remove stop words.

text\_content = [''.join(re.split("[ .,;:!?‘’``''@#$%^\_&\*()<>{}~\n\t\\\-]", word)) for word in text1]

# Create a set of stopwords

stopwords\_wc = set(STOPWORDS)

customised\_words = ['price', 'great'] # If you want to remove any particular word form text which does not contribute much in meaning

new\_stopwords = stopwords\_wc.union(customised\_words)

# Remove stop words

text\_content = [word for word in text\_content if word not in new\_stopwords]

# Take only non-empty entries

text\_content = [s for s in text\_content if len(s) != 0]

# Best to get the lemmas of each word to reduce the number of similar words

text\_content = [WNL.lemmatize(t) for t in text\_content]

nltk\_tokens = nltk.word\_tokenize(text)

bigrams\_list = list(nltk.bigrams(text\_content))

print(bigrams\_list)

dictionary2 = [' '.join(tup) for tup in bigrams\_list]

print (dictionary2)

# Using count vectoriser to view the frequency of bigrams

from sklearn.feature\_extraction.text import CountVectorizer

vectorizer = CountVectorizer(ngram\_range=(2, 2))

bag\_of\_words = vectorizer.fit\_transform(dictionary2)

vectorizer.vocabulary\_

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

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

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

print(words\_freq[:100])

# Generating wordcloud

words\_dict = dict(words\_freq)

WC\_height = 1000

WC\_width = 1500

WC\_max\_words = 200

wordCloud = WordCloud(max\_words=WC\_max\_words, height=WC\_height, width=WC\_width, stopwords=new\_stopwords)

wordCloud.generate\_from\_frequencies(words\_dict)

plt.figure(4)

plt.title('Most frequently occurring bigrams connected by same colour and font size')

plt.imshow(wordCloud, interpolation='bilinear')

plt.axis("off")

plt.show()

