Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454
Number of users: 256,059
Number of products: 74,258
Timespan: Oct 1999 - Oct 2012
Number of Attributes/Columns in data: 10

Attribute Information:

1. Id
2. ProductId - unique identifier for the product
3. UserId - unqiue identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp for the review
9. Summary - brief summary of the review
10. Text - text of the review

Objective:

Given a review, determine whether the review is positive (rating of 4 or 5) or negative (rating of 1 or 2).

[Q] How to determine if a review is positive or negative?

[Ans] We could use Score/Rating. A rating of 4 or 5 can be cosnidered as a positive review. A rating of 1 or 2 can be considered as negative one. A review of rating 3 is considered nuetral and such reviews are ignored from our analysis. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

[1]. Reading Data

[1.1] Loading the data

The dataset is available in two forms

1. .csv file
2. SQLite Database

In order to load the data, We have used the SQLITE dataset as it is easier to query the data and visualise the data efficiently.

Here as we only want to get the global sentiment of the recommendations (positive or negative), we will purposefully ignore all Scores equal to 3. If the score is above 3, then the recommendation wil be set to "positive". Otherwise, it will be set to "negative".

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os
```

```
C:\ProgramData\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWa
rning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_seria
l")
```

In [2]:

```
# using SQLite Table to read data.
con = sqlite3.connect('C:/Users/Excel/Desktop/vins/database.sqlite')

# filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
# SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000, will give top 50
0000 data points
# you can change the number to any other number based on your computing
 power
```

```python
# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Sco
re != 3 LIMIT 500000""", con)
# for tsne assignment you can take 5k data points

filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score
 != 3 LIMIT 50000""", con)

# Give reviews with Score>3 a positive rating(1), and reviews with a sc
ore<3 a negative rating(0).
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

```
Number of data points in our data (50000, 10)
```

Out[2]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulnes
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulnes
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1

In [3]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [4]:
```
print(display.shape)
display.head()
```

(80668, 7)

Out[4]:

	UserId	ProductId	ProfileName	Time	Score	Text	COU
0	#oc-R115TNMSPFT9I7	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2

	UserId	ProductId	ProfileName	Time	Score	Text	COU
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B007Y59HVM	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ET0	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [5]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[5]:

	UserId	ProductId	ProfileName	Time	Score	Text	(
80638	AZY10LLTJ71NX	B006P7E5ZI	undertheshrine "undertheshrine"	1334707200	5	I was recommended to try green tea extract to ...	5

In [6]:
```
display['COUNT(*)'].sum()
```
Out[6]: 393063

[2] Exploratory Data Analysis

[2.1] Data Cleaning: Deduplication

It is observed (as shown in the table below) that the reviews data had many duplicate entries. Hence it was necessary to remove duplicates in order to get unbiased results for the analysis of the data. Following is an example:

In [7]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```
Out[7]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
0	78445	B000HDL1RQ	AR5J8UI46CURR	Geetha Krishnan	2	2

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
1	138317	B000HDOPYC	AR5J8UI46CURR	Geetha Krishnan	2	2
2	138277	B000HDOPYM	AR5J8UI46CURR	Geetha Krishnan	2	2
3	73791	B000HDOPZG	AR5J8UI46CURR	Geetha Krishnan	2	2
4	155049	B000PAQ75C	AR5J8UI46CURR	Geetha Krishnan	2	2

As it can be seen above that same user has multiple reviews with same values for HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary and Text and on doing analysis it was found that

ProductId=B000HDOPZG was Loacker Quadratini Vanilla Wafer Cookies, 8.82-Ounce Packages (Pack of 8)

ProductId=B000HDL1RQ was Loacker Quadratini Lemon Wafer Cookies, 8.82-Ounce Packages (Pack of 8) and so on

It was inferred after analysis that reviews with same parameters other than ProductId belonged to the same product just having different flavour or quantity. Hence in order to reduce redundancy it was decided to eliminate the rows having same parameters.

The method used for the same was that we first sort the data according to ProductId and then just keep the first similar product review and delelte the others. for eg. in the above just the review for ProductId=B000HDL1RQ remains. This method ensures that there is only one representative for each product and deduplication without sorting would lead to possibility of different representatives still existing for the same product.

In [8]:
```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
```

In [9]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first', inplace=False)
final.shape
```

Out[9]:
```
(46072, 10)
```

In [10]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[10]:
```
92.144
```

Observation:- It was also seen that in two rows given below the value of HelpfulnessNumerator is greater than HelpfulnessDenominator which is not practically possible hence these two rows too are removed from calcualtions

In [11]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
```

```
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[11]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2

In [12]:

```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [13]:

```
#Before starting the next phase of preprocessing lets see the number of
 entries left
print(final.shape)

#How many positive and negative reviews are present in our dataset?
final['Score'].value_counts()
```

```
(46071, 10)
```

Out[13]:

```
1    38479
0     7592
```

```
Name: Score, dtype: int64
```

In [14]:
```
final['Time']=pd.to_datetime(final['Time'],unit='s')
final=final.sort_values(by='Time')
```

[3] Preprocessing

[3.1]. Preprocessing Review Text

Now that we have finished deduplication our data requires some preprocessing before we go on further with analysis and making the prediction model.

Hence in the Preprocessing phase we do the following in the order below:-

1. Begin by removing the html tags
2. Remove any punctuations or limited set of special characters like , or . or # etc.
3. Check if the word is made up of english letters and is not alpha-numeric
4. Check to see if the length of the word is greater than 2 (as it was researched that there is no adjective in 2-letters)
5. Convert the word to lowercase
6. Remove Stopwords
7. Finally Snowball Stemming the word (it was obsereved to be better than Porter Stemming)

After which we collect the words used to describe positive and negative reviews

In [15]:
```
# printing some random reviews
sent_0 = final['Text'].values[0]
print(sent_0)
print("="*50)

sent_1000 = final['Text'].values[1000]
print(sent_1000)
print("="*50)
```

```
sent_1500 = final['Text'].values[1500]
print(sent_1500)
print("="*50)

sent_4900 = final['Text'].values[4900]
print(sent_4900)
print("="*50)
```

```
This was a really good idea and the final product is outstanding. I use
the decals on my car window and everybody asks where i bought the decal
s i made.  Two thumbs up!
==================================================
Speaking as another Texan, I think the first rule for these delicious t
reats is to NOT order them during spring or summer. In fact, your safes
t bet is to ONLY order them in the dead of winter.  LOL!  As long as yo
u do that, be prepared for a truly amazing treat!  This package comes w
ith 12 bite-sized delicacies.  The chocolate is high-quality, the nuts
are crunchy, and the overall taste couldn't be better.  Definitely wort
h the price!
==================================================
Fast, easy and definitely delicious.  Makes a great cup of coffee and v
ery easy to make.  Good purchase.  Will continue to order from here.<br
/>Thanx...
==================================================
Naturally this review is based upon my cat's intake of Petite Cuisine.
She's not a particular picky eater, so I can't say much about that. How
ever, she looks to really enjoy this brand of cat food. I have tried so
me brands of wet food in the past that have made her sick (I know cats
seem to have digestive systems that are prone to upsetting!) Petite Cui
sine did not have any effect there, and she really enjoyed all the flav
ors. I don't feed her wet food often, usually just some tuna fish now a
nd then. So, although this food is expensive if you used it at every me
al, it is priced about the same as tuna, so it fits my needs perfectly.
I'm sure my cat will enjoy the rest of this case, and I can keep the ca
nned tuna to myself for now :-)
==================================================
```

In [16]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/40
```

```
84039
sent_0 = re.sub(r"http\S+", "", sent_0)
sent_1000 = re.sub(r"http\S+", "", sent_1000)
sent_150 = re.sub(r"http\S+", "", sent_1500)
sent_4900 = re.sub(r"http\S+", "", sent_4900)

print(sent_0)
```

```
This was a really good idea and the final product is outstanding. I use
the decals on my car window and everybody asks where i bought the decal
s i made.  Two thumbs up!
```

In [17]:

```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how
-to-remove-all-tags-from-an-element
from bs4 import BeautifulSoup

soup = BeautifulSoup(sent_0, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1000, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1500, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_4900, 'lxml')
text = soup.get_text()
print(text)
```

```
This was a really good idea and the final product is outstanding. I use
the decals on my car window and everybody asks where i bought the decal
s i made.  Two thumbs up!
==================================================
Speaking as another Texan, I think the first rule for these delicious t
```

```
reats is to NOT order them during spring or summer. In fact, your safes
t bet is to ONLY order them in the dead of winter.  LOL!  As long as yo
u do that, be prepared for a truly amazing treat!  This package comes w
ith 12 bite-sized delicacies.  The chocolate is high-quality, the nuts
are crunchy, and the overall taste couldn't be better.  Definitely wort
h the price!
==================================================
Fast, easy and definitely delicious.  Makes a great cup of coffee and v
ery easy to make.  Good purchase.  Will continue to order from here.Tha
nx...
==================================================
Naturally this review is based upon my cat's intake of Petite Cuisine.
She's not a particular picky eater, so I can't say much about that. How
ever, she looks to really enjoy this brand of cat food. I have tried so
me brands of wet food in the past that have made her sick (I know cats
seem to have digestive systems that are prone to upsetting!) Petite Cui
sine did not have any effect there, and she really enjoyed all the flav
ors. I don't feed her wet food often, usually just some tuna fish now a
nd then. So, although this food is expensive if you used it at every me
al, it is priced about the same as tuna, so it fits my needs perfectly.
I'm sure my cat will enjoy the rest of this case, and I can keep the ca
nned tuna to myself for now :-)
```

In [18]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
```

```
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [19]:
```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Fast, easy and definitely delicious.  Makes a great cup of coffee and v
ery easy to make.  Good purchase.  Will continue to order from here.<br
/>Thanx...
==================================================
```

In [20]:
```
#remove words with numbers python: https://stackoverflow.com/a/1808237
0/4084039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
This was a really good idea and the final product is outstanding. I use
the decals on my car window and everybody asks where i bought the decal
s i made.  Two thumbs up!
```

In [21]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
Fast easy and definitely delicious Makes a great cup of coffee and very
easy to make Good purchase Will continue to order from here br Thanx
```

In [22]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
# <br /><br /> ==> after the above steps, we are getting "br br"
# we are including them into stop words list
# instead of <br /> if we have <br/> these tags would have revmoved in
 the 1st step

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'o
urs', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselve
```

```
s', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
is', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
ave', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between',
'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'h
ow', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 's
o', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should',
"should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
 "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [23]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(final['Text'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower
```

```
() not in stopwords)
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████████████████████████████| 46071/46071 [00:25<00:00, 180
6.46it/s]
```

In [24]:
```
preprocessed_reviews[1500]
```

Out[24]:
```
'fast easy definitely delicious makes great cup coffee easy make good p
urchase continue order thanx'
```

[3.2] Preprocessing Review Summary

In [25]:
```
## Similartly you can do preprocessing for review summary also.
```

In [26]:
```
final['preprocessed_reviews']=preprocessed_reviews
X=final['preprocessed_reviews'].values
```

In [27]:
```
tf_idf_vect=TfidfVectorizer(min_df=10,max_features=2000)
tf_idf_vect.fit_transform(X)
```

Out[27]:
```
<46071x2000 sparse matrix of type '<class 'numpy.float64'>'
        with 1235447 stored elements in Compressed Sparse Row format>
```

In [28]:
```
topfeatures=tf_idf_vect.get_feature_names()
topfeatures=np.asarray(topfeatures)
```

In [29]:
```
sentence=" ".join(X)
sentence_split=sentence.split(" ")
dic=dict((topfeatures[i],i) for i in range (len(topfeatures)))
```

In [30]:
```
lenw=len(topfeatures)
sentence_length=len(sentence_split)
```

In [31]:
```
#https://github.com/omkar1610/Amazon-Fine-Food-Reviews/blob/master/11%2
```

```
0Amazon%20Fine%20Food%20Reviews%20Analysis_Truncated%20SVD%20(2).pdf
import itertools
def comatrices(windowsize,lenw,sentence_length,dic,sentence_split):

    matrix = np.zeros((lenw,lenw)) # Initializing co occurence matrix
    tmp = sentence_split[0:windowsize+1] # The first window of neigh +
 1 elements.

    for word in tmp:
        if(word in dic):
            loc = dic[word] # Get the location in topword list
            matrix[loc][loc] += 1
    for w1,w2 in itertools.combinations(tmp, 2): # This will give all p
ossible combinations (nC2)
        if((w1 in dic) & (w2 in dic) & (w1 != w2)): # Both must be top
 words and Both must be different
            plac1,plac2 = dic[w1],dic[w2]

            matrix[plac1][plac2] += 1
            matrix[plac2][plac1] += 1


    for i in tqdm(range(1,len(sentence_split) - windowsize)):
        tmp = sentence_split[i:i+windowsize+1] # This is the window of
 neigh + 1 elements
#print(tmp)
        new_word = tmp[windowsize] # Only the last added word is the ne
w one and we have to pair this with other elements
        if(new_word in dic): # only if it is a top word otherwise NO
            loc = dic[new_word] # Location of the new word in topword l
ist
            matrix[loc][loc] += 1 # Because each diag ele is same as th
e no of occurence of the word
            for i in range(windowsize): # pairing new word with other e
lements in the window
                w1, w2 = tmp[i], new_word # getting the location of bot
h words in the topwords list
                if((w1 in dic) & (w1 != new_word)): #if w1 is a topword
 and not same as new word, then we
```

```
                    l1,l2 = dic[w1], loc #get the location in top word
list
                    matrix[l1][l2] += 1
                    matrix[l2][l1] += 1
    return matrix.astype(int)
```

In [32]:
```
matic=comatrices(5,lenw,sentence_length,dic,sentence_split)
```

```
100%|██████████████████████████████| 1797908/1797908 [00:20<00:00, 8741
6.38it/s]
```

In [33]:
```
matic
```

Out[33]:
```
array([[1345,    3,    9, ...,    0,    3,    1],
       [   3,  189,    0, ...,    2,    2,    0],
       [   9,    0, 1322, ...,    3,   10,    0],
       ...,
       [   0,    2,    3, ...,  217,    0,    0],
       [   3,    2,   10, ...,    0,  428,    0],
       [   1,    0,    0, ...,    0,    0,  128]])
```

In [34]:
```
#sanity check of the co-occurence matrix with toy example
A = ["abc def ijk pqr", "pqr klm opq", "lmn pqr xyz abc def pqr abc"]
topfeatures123 = ["abc", "pqr", "def"]
sent1 = " ".join(A)
sent_list1 = sent1.split(" ")
lenw = len(topfeatures123)
dic1 = dict((topfeatures123[i],i) for i in range(lenw))
co_matrix = comatrices(windowsize = 2,lenw=len(topfeatures123),sentence
_length = len(sent_list1),dic=dic1, sentence_split= sent_list1)
co_matrix
```

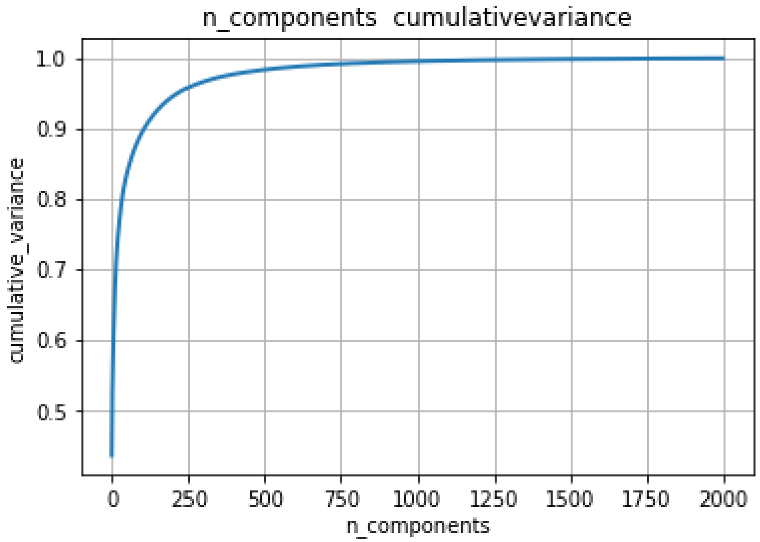
```
100%|████████████████████████████████████████| 11/11 [00:00<00:00, 366
6.35it/s]
```

Out[34]:
```
array([[3, 3, 3],
       [3, 4, 2],
       [3, 2, 2]])
```

In [35]:
```
from sklearn.decomposition import TruncatedSVD
svd=TruncatedSVD(n_components=1999)
svd.fit(matic)
```

Out[35]:
```
TruncatedSVD(algorithm='randomized', n_components=1999, n_iter=5,
       random_state=None, tol=0.0)
```

In [36]:
```
percentage_var_expl=svd.explained_variance_ / np.sum(svd.explained_vari
ance_)
cum_var=np.cumsum(percentage_var_expl)
plt.plot(cum_var,linewidth=2)
plt.grid(True)
plt.xlabel("n_components")
plt.ylabel("cumulative_variance")
plt.title("n_components  cumulativevariance")
plt.show()
```



In [37]:
```
svd=TruncatedSVD(n_components=500)
data=svd.fit_transform(matic)
```

In [38]:
```
from sklearn.cluster import KMeans
```

```
K=[2,3,4,5,7,9,10,11]
inertia_loss=[]
for i in K:
    kmeans=KMeans(n_clusters=i,random_state=0).fit(data)
    inertia_loss.append(kmeans.inertia_)
```

In [39]:
```
plt.plot(K,inertia_loss)
plt.xlabel("k",size=15)
plt.grid(True)
plt.ylabel('inertia_loss',size=15)
plt.title('ERROR PLOT',size=20)
plt.show()
```

In [40]:
```
kmeans=KMeans(n_clusters=4,random_state=0,n_jobs=-1)
kmeans.fit(data)
```

Out[40]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=-1, precompute_distances='auto',
    random_state=0, tol=0.0001, verbose=0)
```

In [41]:
```
cluster1 = []
```

```
cluster2 = []
cluster3 = []
cluster4 = []

for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(preprocessed_reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(preprocessed_reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(preprocessed_reviews[i])
    else :
        cluster4.append(preprocessed_reviews[i])
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
```

```
No. of reviews in Cluster-1 :  34

No. of reviews in Cluster-2 :  1

No. of reviews in Cluster-3 :  1964

No. of reviews in Cluster-4 :  1
```

In [42]:

```
text1=cluster1
text2=cluster2
text3=cluster3
text4=cluster4
```

In [43]:

```
lst=[text1,text2,text3,text4]
for text in lst:
    from wordcloud import WordCloud,STOPWORDS
    stopwords=set(STOPWORDS)



    wordcloud = WordCloud(max_words=10000).generate(str(text))
```

```
    plt.figure(figsize = (15, 15), facecolor = None)
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.show()
```


live
alternative
great
flavor
taste
cardboard
calorie
fill
enough
granola
empty
bars
food
good'
good
candy
dense
excellent


find
well
many
delicious
snack
little
made
good
tea
day
time
try
use
price
cookie
sugar
bought
product
food
brand
sweet
eat
best
tried
coffee
better
healthy
drink
taste
love
dog
store
mix
great
make
chocolate
buy
amazon
nice
still
keep
bag
year
want
one
flavor
add
really
even
treat
much
wonderful
excellent
ingredient
think

1. cluster1 is belong to coffee,great,love,chocolate.
2. cluster2 is belong to bars,food,excellent,flavour.
3. cluster3 is belong to good tea,taste,love,one.
4. cluster4 is belong to love,drink,ice,night,mix.

In [44]:
```
from sklearn.metrics.pairwise import cosine_similarity
similar = cosine_similarity(data,data) # Cosine Similarity Matrix
similar.shape
```

Out[44]: (2000, 2000)

In [45]:
```
def get__word(word):
```

```
        tmp = np.where(topfeatures == word)[0]
        if word in topfeatures:
            index = tmp[0]
            df = pd.DataFrame()
            print(index)
            df['word'] = topfeatures
            df['distance'] = similar[index]
            df = df.sort_values(by = 'distance',kind = 'quicksort',ascendin
g= False)
            print(df.head(10))
        else:
            print("word is not present in topfeatures")
```

In [46]:
```
get__word('tubelight')
```

```
word is not present in topfeatures
```

In [47]:
```
get__word('amazon')
```

```
51
              word  distance
51          amazon  1.000000
341            com  0.963072
1898           via  0.863899
1798          thru  0.856748
1532       selling  0.826018
1342         prime  0.822213
1156        offers  0.816740
1340        prices  0.802991
1704  subscription  0.774399
1533         sells  0.770916
```

In [163]:
```
get__word('like')
```

```
968
            word  distance
```

```
968           like  1.000000
626          feels  0.900749
1632        sounds  0.866245
920          kinda  0.826350
1043      medicine  0.809713
628           felt  0.805494
1253    personally  0.802839
1601       smelled  0.794846
1630          sort  0.765678
1631         sound  0.757205
```

In [164]:
```python
from platform import python_version

print(python_version())
```

```
3.6.5
```

Truncated-SVD

[5.1] Taking top features from TFIDF, SET 2

In [0]:
```python
# Please write all the code with proper documentation
```

[5.2] Calulation of Co-occurrence matrix

In [0]:
```python
# Please write all the code with proper documentation
```

[5.3] Finding optimal value for number of components (n) to be retained.

In [0]:
```python
# Please write all the code with proper documentation
```

[5.4] Applying k-means clustering

In [0]:
```
# Please write all the code with proper documentation
```

[5.5] Wordclouds of clusters obtained in the above section

In [0]:
```
# Please write all the code with proper documentation
```

[5.6] Function that returns most similar words for a given word.

In [0]:
```
# Please write all the code with proper documentation
```

[6] Conclusions

In [0]:
```
# Please write down few lines about what you observed from this assignm
ent.
# Also please do mention the optimal values that you obtained for numbe
r of components & number of clusters.
```