Amazon Fine Food Reviews Analysis

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

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/ (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 [41]:
```
%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
import tabulate

from tqdm import tqdm
import os

from sklearn.externals import joblib
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import math
```

In [2]:
```
# using SQLite Table to read data.
con = sqlite3.connect('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 500000
data points
# you can change the number to any other number based on your computing pow
er

# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score !
= 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 5000""", con)

# Give reviews with Score>3 a positive rating(1), and reviews with a score<
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 (5000, 10)
```

Out[2]:

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

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	Helpful
0	78445	B000HDL1RQ	AR5J8UI46CURR	Geetha Krishnan	2	2
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","Te
xt"}, keep='first', inplace=False)
final.shape
```

Out[9]: (4986, 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]: 99.72

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	Helpful
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 ent
ries left
print(final.shape)

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

```
(4986, 10)
```

Out[13]:
```
1    4178
0     808
Name: Score, dtype: int64
```

[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 [14]:
```
# 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)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /
>http://www.amazon.com/VICTOR-FLY-MAGNET-BAIT-REFILL/dp/B00004RBDY<br /><br
/>The Victor M380 and M502 traps are unreal, of course -- total fly genocid
e. Pretty stinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious these
chips are.  The best thing was that there were a lot of "brown" chips in th
e bsg (my favorite), so I bought some more through amazon and shared with f
amily and friends.  I am a little disappointed that there are not, so far,
very many brown chips in these bags, but the flavor is still very good.  I
like them better than the yogurt and green onion flavor because they do not
seem to be as salty, and the onion flavor is better.  If you haven't eaten
Kettle chips before, I recommend that you try a bag before buying bulk.  Th
ey are thicker and crunchier than Lays but just as fresh out of the bag.
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what they we
re ordering; the other wants crispy cookies.  Hey, I'm sorry; but these rev
iews do nobody any good beyond reminding us to look  before ordering.<br />
<br />These are chocolate-oatmeal cookies.  If you don't like that combinat
ion, don't order this type of cookie.  I find the combo quite nice, really.
The oatmeal sort of "calms" the rich chocolate flavor and gives the cookie
sort of a coconut-type consistency.  Now let's also remember that tastes di
ffer; so, I've given my opinion.<br /><br />Then, these are soft, chewy coo
kies -- as advertised.  They are not "crispy" cookies, or the blurb would s
ay "crispy," rather than "chewy."  I happen to like raw cookie dough; howev
er, I don't see where these taste like raw cookie dough.  Both are soft, ho
wever, so is this the confusion?  And, yes, they stick together.  Soft cook
ies tend to do that.  They aren't individually wrapped, which would add to
the cost.  Oh yeah, chocolate chip cookies tend to be somewhat sweet.<br />
<br />So, if you want something hard and crisp, I suggest Nabiso's Ginger S
naps.  If you want a cookie that's soft, chewy and tastes like a combinatio
n of chocolate and oatmeal, give these a try.  I'm here to place my second
order.
==================================================
love to order my coffee on amazon.  easy and shows up quickly.<br />This k
cup is great coffee.  dcaf is very good as well
==================================================
```

In [15]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/4084039
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)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /
> /><br />The Victor M380 and M502 traps are unreal, of course -- total fly
genocide. Pretty stinky, but only right nearby.
```

In [16]:
```
# 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)
```

```
Why is this $[...] when the same product is available for $[...] here? />Th
e Victor M380 and M502 traps are unreal, of course -- total fly genocide. P
retty stinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious these
chips are.  The best thing was that there were a lot of "brown" chips in th
e bsg (my favorite), so I bought some more through amazon and shared with f
amily and friends.  I am a little disappointed that there are not, so far,
very many brown chips in these bags, but the flavor is still very good.  I
like them better than the yogurt and green onion flavor because they do not
seem to be as salty, and the onion flavor is better.  If you haven't eaten
Kettle chips before, I recommend that you try a bag before buying bulk.  Th
ey are thicker and crunchier than Lays but just as fresh out of the bag.
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what they we
re ordering; the other wants crispy cookies.  Hey, I'm sorry; but these rev
iews do nobody any good beyond reminding us to look  before ordering.These
are chocolate-oatmeal cookies.  If you don't like that combination, don't o
rder this type of cookie.  I find the combo quite nice, really.  The oatmea
l sort of "calms" the rich chocolate flavor and gives the cookie sort of a
coconut-type consistency.  Now let's also remember that tastes differ; so,
I've given my opinion.Then, these are soft, chewy cookies -- as advertised.
They are not "crispy" cookies, or the blurb would say "crispy," rather than
"chewy."  I happen to like raw cookie dough; however, I don't see where the
se taste like raw cookie dough.  Both are soft, however, so is this the con
fusion?  And, yes, they stick together.  Soft cookies tend to do that.  The
y aren't individually wrapped, which would add to the cost.  Oh yeah, choco
late chip cookies tend to be somewhat sweet.So, if you want something hard
and crisp, I suggest Nabiso's Ginger Snaps.  If you want a cookie that's so
ft, chewy and tastes like a combination of chocolate and oatmeal, give thes
e a try.  I'm here to place my second order.
==================================================
love to order my coffee on amazon.  easy and shows up quickly.This k cup is
great coffee.  dcaf is very good as well
```

In [17]:

```python
# 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 [18]:
```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Wow.  So far, two two-star reviews.  One obviously had no idea what they we
re ordering; the other wants crispy cookies.  Hey, I am sorry; but these re
views do nobody any good beyond reminding us to look  before ordering.<br /
><br />These are chocolate-oatmeal cookies.  If you do not like that combin
ation, do not order this type of cookie.  I find the combo quite nice, real
ly.  The oatmeal sort of "calms" the rich chocolate flavor and gives the co
okie sort of a coconut-type consistency.  Now let is also remember that tas
tes differ; so, I have given my opinion.<br /><br />Then, these are soft, c
hewy cookies -- as advertised.  They are not "crispy" cookies, or the blurb
would say "crispy," rather than "chewy."  I happen to like raw cookie doug
h; however, I do not see where these taste like raw cookie dough.  Both are
soft, however, so is this the confusion?  And, yes, they stick together.  S
oft cookies tend to do that.  They are not individually wrapped, which woul
d add to the cost.  Oh yeah, chocolate chip cookies tend to be somewhat swe
et.<br /><br />So, if you want something hard and crisp, I suggest Nabiso i
s Ginger Snaps.  If you want a cookie that is soft, chewy and tastes like a
combination of chocolate and oatmeal, give these a try.  I am here to place
my second order.
==================================================
```

In [19]:
```
#remove words with numbers python: https://stackoverflow.com/a/18082370/408
4039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /
> /><br />The Victor  and  traps are unreal, of course -- total fly genocid
e. Pretty stinky, but only right nearby.
```

In [20]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
Wow So far two two star reviews One obviously had no idea what they were or
dering the other wants crispy cookies Hey I am sorry but these reviews do n
obody any good beyond reminding us to look before ordering br br These are
chocolate oatmeal cookies If you do not like that combination do not order
this type of cookie I find the combo quite nice really The oatmeal sort of
calms the rich chocolate flavor and gives the cookie sort of a coconut type
consistency Now let is also remember that tastes differ so I have given my
opinion br br Then these are soft chewy cookies as advertised They are not
crispy cookies or the blurb would say crispy rather than chewy I happen to
like raw cookie dough however I do not see where these taste like raw cooki
e dough Both are soft however so is this the confusion And yes they stick t
ogether Soft cookies tend to do that They are not individually wrapped whic
h would add to the cost Oh yeah chocolate chip cookies tend to be somewhat
sweet br br So if you want something hard and crisp I suggest Nabiso is Gin
ger Snaps If you want a cookie that is soft chewy and tastes like a combina
tion of chocolate and oatmeal give these a try I am here to place my second
order
```

In [21]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# <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', 'ours'
, 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have'
, 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'be
cause', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'int
o', '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', 'how',
'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so',
'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "sho
uld've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'did
n', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sh
ouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [22]:
```
# 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() no
t in stopwords)
    preprocessed_reviews.append(sentance.strip())
```

```
100%|
| 4986/4986 [00:01<00:00, 3137.37it/s]
```

In [23]:
```
preprocessed_reviews[1500]
```

Out[23]:
```
'wow far two two star reviews one obviously no idea ordering wants crispy c
ookies hey sorry reviews nobody good beyond reminding us look ordering choc
olate oatmeal cookies not like combination not order type cookie find combo
quite nice really oatmeal sort calms rich chocolate flavor gives cookie sor
t coconut type consistency let also remember tastes differ given opinion so
ft chewy cookies advertised not crispy cookies blurb would say crispy rathe
r chewy happen like raw cookie dough however not see taste like raw cookie
dough soft however confusion yes stick together soft cookies tend not indiv
idually wrapped would add cost oh yeah chocolate chip cookies tend somewhat
sweet want something hard crisp suggest nabiso ginger snaps want cookie sof
t chewy tastes like combination chocolate oatmeal give try place second ord
er'
```

[3.2] Preprocessing Review Summary

In [6]:
```
## Similarly you can do preprocessing for review summary also.
```

[4] Featurization

[4.1] BAG OF WORDS

In [25]:
```
#BoW
count_vect = CountVectorizer() #in scikit-learn
count_vect.fit(preprocessed_reviews)
print("some feature names ", count_vect.get_feature_names()[:10])
print('='*50)

final_counts = count_vect.transform(preprocessed_reviews)
print("the type of count vectorizer ",type(final_counts))
print("the shape of out text BOW vectorizer ",final_counts.get_shape())
print("the number of unique words ", final_counts.get_shape()[1])
```

```
some feature names  ['aa', 'aahhhs', 'aback', 'abandon', 'abates', 'abbot
t', 'abby', 'abdominal', 'abiding', 'ability']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (4986, 12997)
the number of unique words  12997
```

[4.2] Bi-Grams and n-Grams.

In [26]:
```
#bi-gram, tri-gram and n-gram

#removing stop words like "not" should be avoided before building n-grams
# count_vect = CountVectorizer(ngram_range=(1,2))
# please do read the CountVectorizer documentation http://scikit-learn.org/
stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.ht
ml

# you can choose these numebrs min_df=10, max_features=5000, of your choice
count_vect = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=500
0)
final_bigram_counts = count_vect.fit_transform(preprocessed_reviews)
print("the type of count vectorizer ",type(final_bigram_counts))
print("the shape of out text BOW vectorizer ",final_bigram_counts.get_shape
())
print("the number of unique words including both unigrams and bigrams ", fi
nal_bigram_counts.get_shape()[1])
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (4986, 3144)
the number of unique words including both unigrams and bigrams  3144
```

[4.3] TF-IDF

In [27]:
```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
tf_idf_vect.fit(preprocessed_reviews)
print("some sample features(unique words in the corpus)",tf_idf_vect.get_fe
ature_names()[0:10])
print('='*50)

final_tf_idf = tf_idf_vect.transform(preprocessed_reviews)
print("the type of count vectorizer ",type(final_tf_idf))
print("the shape of out text TFIDF vectorizer ",final_tf_idf.get_shape())
print("the number of unique words including both unigrams and bigrams ", fi
nal_tf_idf.get_shape()[1])
```

```
some sample features(unique words in the corpus) ['ability', 'able', 'able
find', 'able get', 'absolute', 'absolutely', 'absolutely delicious', 'absol
utely love', 'absolutely no', 'according']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (4986, 3144)
the number of unique words including both unigrams and bigrams  3144
```

[4.4] Word2Vec

In [28]:
```
# Train your own Word2Vec model using your own text corpus
i=0
list_of_sentance=[]
for sentance in preprocessed_reviews:
    list_of_sentance.append(sentance.split())
```

In [42]:
```
# Using Google News Word2Vectors

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file wich contains a dict ,
# and it contains all our courpus words as keys and  model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.


# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17SRFA
zZPY
# you can comment this whole cell
# or change these varible according to your need

is_your_ram_gt_16g=False
want_to_use_google_w2v = False
want_to_train_w2v = True

if want_to_train_w2v:
    # min_count = 5 considers only words that occured atleast 5 times
    w2v_model=Word2Vec(list_of_sentance,min_count=5,size=50, workers=4)
    print(w2v_model.wv.most_similar('great'))
    print('='*50)
    print(w2v_model.wv.most_similar('worst'))

elif want_to_use_google_w2v and is_your_ram_gt_16g:
    if os.path.isfile('GoogleNews-vectors-negative300.bin'):
        w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-neg
ative300.bin', binary=True)
        print(w2v_model.wv.most_similar('great'))
        print(w2v_model.wv.most_similar('worst'))
    else:
        print("you don't have gogole's word2vec file, keep want_to_train_w2
v = True, to train your own w2v ")
```

```
[('snack', 0.9951335191726685), ('calorie', 0.9946465492248535), ('wonderfu
l', 0.9946032166481018), ('excellent', 0.9944332838058472), ('especially',
0.9941144585609436), ('baked', 0.9940600395202637), ('salted', 0.9940472245
21637), ('alternative', 0.9937226176261902), ('tasty', 0.9936816692352295),
('healthy', 0.9936649799346924)]
==================================================
[('varieties', 0.9994194507598877), ('become', 0.9992934465408325), ('popco
rn', 0.9992750883102417), ('de', 0.9992610216140747), ('miss', 0.9992451071
739197), ('melitta', 0.999218761920929), ('choice', 0.9992102384567261),
('american', 0.9991837739944458), ('beef', 0.9991780519485474), ('finish',
0.9991567134857178)]
```

In [36]:
```
w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 5 times  3817
sample words  ['product', 'available', 'course', 'total', 'pretty', 'stink
y', 'right', 'nearby', 'used', 'ca', 'not', 'beat', 'great', 'received', 's
hipment', 'could', 'hardly', 'wait', 'try', 'love', 'call', 'instead', 'rem
oved', 'easily', 'daughter', 'designed', 'printed', 'use', 'car', 'window
s', 'beautifully', 'shop', 'program', 'going', 'lot', 'fun', 'everywhere',
'like', 'tv', 'computer', 'really', 'good', 'idea', 'final', 'outstanding',
'window', 'everybody', 'asks', 'bought', 'made']
```

[4.4.1] Converting text into vectors using Avg W2V, TFIDF-W2V

[4.4.1.1] Avg W2v

In [38]:
```
# average Word2Vec
# compute average word2vec for each review.
sent_vectors = []; # the avg-w2v for each sentence/review is stored in this
list
for sent in tqdm(list_of_sentance): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, you mi
ght need to change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors.append(sent_vec)
print(len(sent_vectors))
print(len(sent_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████
└└└
```

In [41]:
```
# TF-IDF weighted Word2Vec
tfidf_feat = model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and cell_v
al = tfidf

tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored
in this list
row=0;
for sent in tqdm(list_of_sentance): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/revie
w
    for word in sent: # for each word in a review/sentence
        if word in w2v_words and word in tfidf_feat:
            vec = w2v_model.wv[word]
#             tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
            # to reduce the computation we are
            # dictionary[word] = idf value of word in whole courpus
            # sent.count(word) = tf valeus of word in this review
            tf_idf = dictionary[word]*(sent.count(word)/len(sent))
            sent_vec += (vec * tf_idf)
            weight_sum += tf_idf
    if weight_sum != 0:
        sent_vec /= weight_sum
    tfidf_sent_vectors.append(sent_vec)
    row += 1
```

```
100%|
| 4986/4986 [00:20<00:00, 245.63it/s]
```

[5] Assignment 10: K-Means, Agglomerative & DBSCAN Clustering

1. **Apply K-means Clustering on these feature sets:**

   - SET 1:Review text, preprocessed one converted into vectors using (BOW)
   - SET 2:Review text, preprocessed one converted into vectors using (TFIDF)
   - SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
   - SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)
   - Find the best 'k' using the elbow-knee method (plot k vs inertia_)
   - Once after you find the k clusters, plot the word cloud per each cluster so that at a single go we can analyze the words in a cluster.

2. **Apply Agglomerative Clustering on these feature sets:**

   - SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
   - SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)
   - Apply agglomerative algorithm and try a different number of clusters like 2,5 etc.
   - Same as that of K-means, plot word clouds for each cluster and summarize in your own words what that cluster is representing.
   - You can take around 5000 reviews or so(as this is very computationally expensive one)

3. **Apply DBSCAN Clustering on these feature sets:**

   - SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
   - SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)
   - Find the best 'Eps' using the elbow-knee method. (https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r/48558030#48558030)
   - Same as before, plot word clouds for each cluster and summarize in your own words what that cluster is representing.
   - You can take around 5000 reviews for this as well.

[5.1] K-Means Clustering

Code for Vectorizers

In [ ]:
```
#%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np


import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report,accuracy_score,confusion_
matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve,auc
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE


from bs4 import BeautifulSoup
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
import nltk
from nltk import word_tokenize, sent_tokenize
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
import time

class asign_10_dtcrea:

        def __init__(self):
                print('In the constructor\n')
                self.X_train = pd.DataFrame()  #for gridsearchcv

        #gridsearchcv parameters -- start
```

```
        @property
        def X_train(self):
            return self._X_train

        @X_train.setter
        def X_train(self,new_X_train):
            self._X_train = new_X_train


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

        def write_ft_data(self,fnme,opdata):

                #fname = 'E:/appliedaicourse/assignments/dblite/kdtree_50
k/' + fnme
                fname = 'E:/appliedaiacourse/assignments/dblite/asign-10-cl
uster' + fnme
                with open(fname, 'wb') as fp:
                        pickle.dump(opdata, fp)

        def write_data(self,fnme,opdata):

                #fname = 'E:/appliedaicourse/assignments/dblite/kdtree_50
k/' + fnme

                fname = 'E:/appliedaiacourse/assignments/dblite/asign-10-cl
uster' + fnme
                print(fname)
                with open(fname, 'wb') as fp:
                        pickle.dump(opdata, fp)


        def decontracted(self,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



        # Combining all the above statements
```

```
        def rw_preproc(self,xdata):
                stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'w
e', 'our', 'ours', '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"])

                preprocessed_reviews = []

                # tqdm is for printing the status bar

                for sentance in tqdm(xdata.values):
                        sentance = re.sub(r"http\S+", "", sentance)
                        sentance = BeautifulSoup(sentance, 'lxml').get_text
()
                        sentance = self.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.spl
it() if e.lower() not in stopwords)
                        preprocessed_reviews.append(sentance.strip())
                return preprocessed_reviews


        def getreviews(self, nrows):
                X_trn = pd.DataFrame()
                # using SQLite Table to read data.
                filepath = os.path.abspath('E:/appliedaiacourse/assignment
s/dblite/database.sqlite')
                assert os.path.exists(filepath), 'the file does not exist'
                con = sqlite3.connect(filepath)
                #filtered_data = pd.read_sql_query(""" SELECT * FROM Review
```

```
s WHERE Score != 3 LIMIT 50000""", con)
                if nrows == -1 :
                        # fetch all rows
                        filtered_data = pd.read_sql_query(""" SELECT * FROM
Reviews WHERE Score != 3 """ , con)
                else:

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

                #changing reviews with score less than 3 to be positive and
vice-versa
                actualScore = filtered_data['Score']
                positiveNegative = actualScore.map(self.partition)
                filtered_data['Score'] = positiveNegative
                #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')
                #Deduplication of entries
                final=sorted_data.drop_duplicates(subset={"UserId","Profile
Name","Time","Text"}, keep='first', inplace=False)
                final.shape

                final=final[final.HelpfulnessNumerator<=final.HelpfulnessDe
nominator]

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

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

                X_trn, X_tst, self.y_train, self.y_test = train_test_split(
final['Text'], final['Score'], stratify= final['Score'],test_size=0.2, rand
om_state=42)
                xtrn, xval, self.ytrain, self.yval   = train_test_split(X_t
rn, self.y_train, test_size=0.2, random_state=42)

                self.X_train = self.rw_preproc(final['Text'])




        def BOWVectorizer(self):
                X_v_train = []
                bowXtrain_scaled = []

                #this is for gridsearchcv
                count_vect_1 = CountVectorizer(max_features=2000) #in sciki
t-learn
                X_v_train = count_vect_1.fit_transform(self.X_train)


                scaler_1 = StandardScaler(with_mean=False)
                bowXtrain_scaled = scaler_1.fit_transform(X_v_train)
```

```
                self.write_data('/bowvectorizer/ppvectscld_x_train',bowXtra
in_scaled)
                self.write_ft_data('/bowvectorizer/bow_feat',count_vect_1.g
et_feature_names())
                print('sample features: ',count_vect_1.get_feature_names()[
0:50] )


        def TFIDFVectorizer(self):
                X_v_train = []
                tidfXtrain_scaled = []

                #this is for gridsearchcv
                tf_idf_vect_1 = TfidfVectorizer(ngram_range=(1,2), min_df=1
0,max_features=2000) #in scikit-learn
                X_v_train = tf_idf_vect_1.fit_transform(self.X_train)

                scaler_1 = StandardScaler(with_mean=False)
                tidfXtrain_scaled = scaler_1.fit_transform(X_v_train.toarra
y())

                self.write_data('/tfidfvectorizer/ppvectscld_x_train',tidfX
train_scaled)


                self.write_ft_data('/tfidfvectorizer/tfidf_feat',tf_idf_vec
t_1.get_feature_names())
                print("some sample features(unique words in the corpus)",tf
_idf_vect_1.get_feature_names()[0:10])
                print('='*50)

        # average Word2Vec
        # compute average word2vec for each review.
        def w2vec_crea(self, list_of_sentance,w2v_model,w2v_words):
                sent_vectors = []; # the avg-w2v for each sentence/review i
s stored in this list
                for sent in tqdm(list_of_sentance): # for each review/sente
nce
                        sent_vec = np.zeros(2000) # as word vectors are of
 zero length 50, you might need to change this to 300 if you use google's w
2v
                        cnt_words =0; # num of words with a valid vector in
the sentence/review
                        for word in sent: # for each word in a review/sente
nce
                                if word in w2v_words:
                                        vec = w2v_model.wv[word]
                                        sent_vec += vec
                                        cnt_words += 1
                        if cnt_words != 0:
                                sent_vec /= cnt_words
                        sent_vectors.append(sent_vec)

                #print(sent_vectors[0],len(sent_vectors[0]))
                print('w2vec_crea')
                #val=input('w2vecrea')
```

```
                return sent_vectors

        def tfidfwtw2v_crea(self,tfidf_feat, list_of_sentance, w2v_model,w2
v_words,diction,fnme):
            tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/revi
ew is stored in this list
            row=0;
            time1 = time.time()
            svec_reinit = np.zeros(50)
            for sent in tqdm(list_of_sentance): # for each review/sentence
                sent_vec = np.zeros(50) # as word vectors are of zero lengt
h
                weight_sum =0; # num of words with a valid vector in the se
ntence/review
                for word in sent: # for each word in a review/sentence
                    if ((word in w2v_words) and (word in tfidf_feat) and (w
ord in diction)):
                        vec = w2v_model.wv[word]
            #             tf_idf = tf_idf_matrix[row, tfidf_feat.index(wor
d)]
                        # to reduce the computation we are
                        # dictionary[word] = idf value of word in whole cou
rpus
                        # sent.count(word) = tf valeus of word in this revi
ew

                        #print(diction[word],sent.count(word),len(sent))
                        denom = sent.count(word)/len(sent)

                        tf_idf = diction[word]*(denom)

                        sent_vec += (vec * tf_idf)
                        weight_sum += tf_idf
                if weight_sum != 0:
                    sent_vec /= weight_sum
                tfidf_sent_vectors.append(sent_vec)
                print('tfidfwtw2v_crea',tfidf_sent_vectors)
                #val=input('tfidfw2vecrea')
                row += 1
            print("time takes for function \t {0}\n".format(time.time() - t
ime1))
            return tfidf_sent_vectors



        def word2vectorizer(self):

                listsent_X_train=[]

                for sentance in self.X_train : #preprocessed_reviews:
                        listsent_X_train.append(sentance.split())


                # min_count = 5 considers only words that occured atleast 5
times
                #w2v_mdl_X_train=Word2Vec(listsent_X_train,min_count=5,size
=50, workers=4)
```

```
                w2v_mdl_X_train=Word2Vec(listsent_X_train,min_count=5,size=
2000, workers=4)
                print(w2v_mdl_X_train.wv.most_similar('great'))
                print('='*50)
                print(w2v_mdl_X_train.wv.most_similar('worst'))

                w2v_words_X_train = list(w2v_mdl_X_train.wv.vocab)


                w2v_X_train = self.w2vec_crea(listsent_X_train,w2v_mdl_X_tr
ain,w2v_words_X_train)

                scaler1 = StandardScaler()
                avgw2vXtrain_scaled = scaler1.fit_transform(w2v_X_train)



                self.write_data('/avgw2vectorizer/ppvectscld_x_train',avgw2
vXtrain_scaled)
                self.write_data('/avgw2vectorizer/w2vec_feat',w2v_words_X_t
rain)


                """
                        this part is for the tfidf weighted word2vec. Most
 of the above code is common
                        for this part so we will write that code here
                """
                """
                #this is for gridsearchcv
                tf_idf_vect_1 = TfidfVectorizer(ngram_range=(1,2), min_df=1
0,max_features=500) #in scikit-learn
                X_v_train = tf_idf_vect_1.fit_transform(self.X_train)


                dict_X_train = dict(zip(tf_idf_vect_1.get_feature_names(),
 list(X_v_train[0,:].nonzero()[1])))

                tfidf_feat_1 = tf_idf_vect_1.get_feature_names()

                tfidf_avgw2vXtrain = self.tfidfwtw2v_crea(tfidf_feat_1,list
sent_X_train,w2v_mdl_X_train,w2v_words_X_train,dict_X_train,'sent_vect_xtra
in')

                self.write_data('/tfidfwtw2vectorizer/ppvectscld_x_train',t
fidf_avgw2vXtrain)

                self.write_ft_data('/tfidfwtw2vectorizer/tfidf_feat',tf_idf
_vect_1.get_feature_names())
                print("some sample features(unique words in the corpus)",tf
_idf_vect_1.get_feature_names()[0:10])
                print('='*50)
                """
        def writeTfidfVetorizer(self):
                #this is for gridsearchcv
                tf_idf_vect_1 = TfidfVectorizer(ngram_range=(1,2), min_df=1
0,max_features=500) #in scikit-learn
```

```
                X_v_train = tf_idf_vect_1.fit_transform(self.X_train)
                X_v_test = tf_idf_vect_1.transform(self.X_test)
                tfidf_feat_1 = tf_idf_vect_1.get_feature_names()


if __name__ == "__main__" :
        print('coming into main')
        cluster = asign_10_dtcrea()

        cluster.getreviews(5000)

        cluster.BOWVectorizer()
        #cluster.TFIDFVectorizer()
        #cluster.word2vectorizer()
```

[5.1.1] Applying K-Means Clustering on BOW, SET 1

In [3]:
```
# Please write all the code with proper documentation
```

Code for K-Means Clustering on BOW

In [ ]:
```
import numpy as np
import pandas as pd
import pickle
import math

from datetime import datetime
from sklearn.cluster import KMeans
from sklearn.externals import joblib

class asign10tsk1 :

        def __init__(self):
                self.X_train =pd.DataFrame()
                self.ipfname = ''
                self.opfname = ''

                self.los_val = []
                self.nclusters = []

                self.optkclst = None
                self.kmeans = None
                self.labels = None


                self.bowfeat = []
                self.bowwrdcld = []

                self.tfidfeat = []
                self.tfidfwrdcld = []

                self.w2vecfeat = []
                self.w2vecwrdcld = []

        def read_bow_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/bowvectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'bow_feat', 'rb') as fp:
                        self.bowfeat = pickle.load(fp)

        def read_tfidf_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/tfidfvectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'tfidf_feat', 'rb') as fp:
                        self.tfidfeat = pickle.load(fp)

        def read_avgw2vec_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/avgw2vectorizer/'
```

```
                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'w2vec_feat', 'rb') as fp:
                        self.w2vecfeat = pickle.load(fp)

        def read_tfidfwtw2vec_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/tfidfwtw2vectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'tfidf_feat', 'rb') as fp:
                        self.tfidfeat = pickle.load(fp)

        def findkinkmeans(self,mink,maxk):

                self.nclusters = [x for x in range(mink,maxk)]
                print(self.nclusters)
                start_tme = datetime.now()
                print("findkinkmeans process starts",start_tme)
                for i in self.nclusters:

                        self.kmeans = KMeans(n_clusters=i,n_jobs=3, random_
state=42, verbose=200)
                        self.kmeans.fit(self.X_train)
                        self.los_val.append(self.kmeans.inertia_)
                        joblib.dump(self.los_val,self.opfname+'lossval.pkl'
)
                        joblib.dump(i,self.opfname+'curr_nclust.pkl')
                        print('Finished processing cluster:#', i ,' of: ',m
axk, ' Now: ', datetime.now(), ' Duration: ', datetime.now()-start_tme)

                self.optkclst = np.argmin(self.los_val)
                print(self.optkclst)

        def bowkmeans(self):
                #since there is no clear elbow reading the value from the g
raph
                #and setting to 20
                self.kmeans = KMeans(n_clusters=20,n_jobs=3, random_state=4
2, verbose=200)
                self.kmeans.fit(self.X_train)
                self.labels = self.kmeans.labels_

                bowlblftnme = sorted(zip(self.labels,self.bowfeat))

                self.bowrdcld = [word for word in bowlblftnme]
                df = pd.DataFrame(self.bowrdcld, columns =['lbl', 'ftname'
])
                joblib.dump(self.labels,self.opfname+'bowlabels.pkl')
                joblib.dump(self.bowrdcld,self.opfname+'bowrdcld.pkl')
                joblib.dump(df,self.opfname+'df_bowlblftname.pkl')

        def tfidfkmeans(self):
```

```
                self.kmeans = KMeans(n_clusters=self.optkclst,n_jobs=3, ran
dom_state=42, verbose=200)
                self.kmeans.fit(self.X_train)
                self.labels = self.kmeans.labels_

                tfidflblftnme = sorted(zip(self.labels,self.tfidfeat))

                self.tfidfwrdcld = [word for word in tfidflblftnme]
                df = pd.DataFrame(self.tfidfwrdcld, columns =['lbl', 'ftnam
e'])
                joblib.dump(self.labels,self.opfname+'tfidflabels.pkl')
                joblib.dump(self.tfidfwrdcld,self.opfname+'tfidfwrdcld.pkl'
)
                joblib.dump(df,self.opfname+'df_tfidflblftname.pkl')

        def avgw2veckmeans(self):
                #since there is no clear elbow reading the value from the g
raph
                #and setting to 20
                self.kmeans = KMeans(n_clusters=20,n_jobs=3, random_state=4
2, verbose=200)
                self.kmeans.fit(self.X_train)
                self.labels = self.kmeans.labels_

                avgw2vlblftnme = sorted(zip(self.labels,self.w2vecfeat))

                self.w2vecwrdcld = [word for word in avgw2vlblftnme]
                df = pd.DataFrame(self.w2vecwrdcld, columns =['lbl', 'ftnam
e'])
                joblib.dump(self.labels,self.opfname+'avgw2vlabels.pkl')
                joblib.dump(self.w2vecwrdcld,self.opfname+'avgw2vwrdcld.pk
l')
                joblib.dump(df,self.opfname+'df_avgw2vlblftname.pkl')

        def tfidfwtw2veckmeans(self):
                # reading the value from the elbow plot and setting
                # n_cluster to 6
                self.kmeans = KMeans(n_clusters=6,n_jobs=3, random_state=42
, verbose=200)
                self.kmeans.fit(self.X_train)
                self.labels = self.kmeans.labels_

                wtw2vlblftnme = sorted(zip(self.labels,self.tfidfeat))

                self.tfidfwrdcld = [word for word in wtw2vlblftnme]
                df = pd.DataFrame(self.tfidfwrdcld, columns =['lbl', 'ftnam
e'])
                joblib.dump(self.labels,self.opfname+'wtw2vlabels.pkl')
                joblib.dump(self.tfidfwrdcld,self.opfname+'wtw2vwrdcld.pkl'
)
                joblib.dump(df,self.opfname+'df_wtw2vlblftname.pkl')

if __name__ == "__main__":


        bow = asign10tsk1()
        bow.read_bow_dt()
```

```
        bow.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/bo
w/'
        print(bow.X_train.shape)
        #bow.X_train = bow.X_train[:500]
        mink = 2
        #maxk= int(math.sqrt(bow.X_train.shape[0]/2)+1.0)
        maxk= int(math.sqrt(bow.X_train.shape[1]/2)+1.0)
        print(maxk)
        bow.findkinkmeans(mink, maxk)
        bow.bowkmeans()
        del bow

        tfidf =asign10tsk1()
        tfidf.read_tfidf_dt()
        print(tfidf.X_train.shape)
        tfidf.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
fidf/'
        #tfidf.X_train = tfidf.X_train[:500]
        mink = 2
        maxk= int(math.sqrt(tfidf.X_train.shape[1]/2)+1.0)
        print(maxk)
        tfidf.findkinkmeans(mink, maxk)
        tfidf.tfidfkmeans()
        del tfidf

        aw2v =asign10tsk1()
        aw2v.read_avgw2vec_dt()
        print(aw2v.X_train.shape)
        aw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/av
gw2vec/'
        #tfidf.X_train = tfidf.X_train[:500]
        mink = 2
        maxk= int(math.sqrt(aw2v.X_train.shape[1]/2)+1.0)
        print(maxk)
        aw2v.findkinkmeans(mink, maxk)
        aw2v.avgw2veckmeans()
        del aw2v


        wtw2v =asign10tsk1()
        wtw2v.read_tfidfwtw2vec_dt()
        #print(wtw2v.X_train.shape)
        wtw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
fidfwtw2vec/'

        mink = 2
        maxk= int(math.sqrt(len(wtw2v.X_train)/2)+1.0)
        print(maxk)
        wtw2v.findkinkmeans(mink, maxk)
        wtw2v.tfidfwtw2veckmeans()
```

In [34]:
```
los_val= joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/bow/
lossval.pkl')
mink= 2
maxk= int(math.sqrt(2000/2)+1.0)
nclusters = [x for x in range(mink,maxk)]
```

In [35]:
```
print(nclusters,len(los_val))
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31] 30
```

In [40]:
```
fig, ax = plt.subplots(figsize=(16,16))
ax.plot(nclusters, los_val,c='g')
for i, txt in enumerate(np.round(los_val,3)):
    ax.annotate((nclusters[i],np.round(txt,3)), (nclusters[i],los_val[i]))
plt.grid()
plt.title("Loss value for each K")
plt.xlabel("K's")
plt.ylabel("Error measure")
plt.show()
```

[5.1.2] Wordclouds of clusters obtained after applying k-means on BOW SET 1

In [3]:
```
# Please write all the code with proper documentation
```

In [2]:
```
bowlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/b
ow/bowlabels.pkl')
bowrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/bo
w/bowrdcld.pkl')
df = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/bow/df_b
owlblftname.pkl')
```

In [3]:
```
print(bowlabels)
print(np.unique(bowlabels))
optk = np.unique(bowlabels).shape[0]
```

```
[2 2 2 ... 2 2 2]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [7]:
```
for i in range(optk):
    a= list(df[df.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)
    if len(str1) > 0:
        if i == 2 or i == 10:
            plt.figure(figsize=(16,16))
            wordcld = WordCloud(width=1000, height=1000, background_color =
'white',min_font_size=14).generate(str1)
        else:
            plt.figure(figsize=(10,10))
            wordcld = WordCloud(width=500, height=500, background_color =
'white',min_font_size=10).generate(str1)
        plt.imshow(wordcld)
```

use
0
100
200
300
400
0
100
200
300
400

certified
chance
0
100
200
300
400
0
100
200
300
400

adult
add
almond
allow
bean
apple
amount
bag
chocolate
brownie
animal
calorie
bottle
benefit
chemical
case
bill
carb
bring
boy
cat
chip
choice
cake
come
brand
beverage
coffee
cookie
customer
cook
meal
clam
effect
dog
issue
result
cost
color
fruit
complaint
herb
love
ingredient
container
eat
eater
reviewer
pound
hour
hand
piece
keep
sound
sugar
enjoy
newman
onion
pet
oil
leave
cup
ounce
slice
pancake
grain
store
muffin
shot
show
live
say
gift
run
think
plant
year
stay
salad
pack
soup
problem
work
way
supermarket
roll
item
food
oreo
morning
remind
serving
make
point
happen
provide
pod
stick
seed
kind
sweetener
state
fat
help
require
lot
minute
date
recipe
picture
thank
thing
lover
hole
kid
drink
take
taste
save
vitamin
organic
sauce
want
look
review
level
size
sell
nut
know
0
200
400
600
800
1000

0
100
200
300
400
peanuts
0
100
200
300
400

expecting
energy
enjoyed
expensive
0
100
200
300
400
0
100
200
300
400

0
100
200
300
400
avoid
0
100
200
300
400

blend
bit
carrot
favorite
egg
seasoning
feel
time
allergies
active
added
brew
blue
bbq
basket
basil
brewer
bland
bad
addicting
tea
bacon
turn
beware
barbeque
awful
bars
box
cents
calls
biscuit
actually
cause
carried
batch
contained
certainly
bites
elsewhere
away
bitter
cafe
fridge
coat
consume
contacted
fence
digest
crackers
aware
bodied
baby
cilantro
figure
dollar
crisps
flavor
flaxseeds
0
200
400
600
800
1000

topping
0
100
200
300
400
0
100
200
300
400

key
iron
fans
website
wild
fans
whenever
0
100
200
300
400
0
100
200
300
400

pork
monthly
addicted
vegetable
drinker
earth
due
mustard
second
dinner
months
0
100
200
300
400
0
100
200
300
400

[5.1.3] Applying K-Means Clustering on TFIDF, SET 2

In [3]:
```
# Please write all the code with proper documentation
```

In [ ]:
```
        tfidf =asign10tsk1()
        tfidf.read_tfidf_dt()
        print(tfidf.X_train.shape)
        tfidf.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
fidf/'
        #tfidf.X_train = tfidf.X_train[:500]
        mink = 2
        maxk= int(math.sqrt(tfidf.X_train.shape[1]/2)+1.0)
        print(maxk)
        tfidf.findkinkmeans(mink, maxk)
        tfidf.tfidfkmeans()
        del tfidf
```

[5.1.4] Wordclouds of clusters obtained after applying k-means on TFIDF SET 2

In [3]:
```
# Please write all the code with proper documentation
```

In [3]:
```
los_val= joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tfid
f/lossval.pkl')
mink= 2
maxk= int(math.sqrt(2000/2)+1.0)
nclusters = [x for x in range(mink,maxk)]
```

In [4]:
```
print(nclusters,len(los_val))
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31] 30
```

In [5]:
```
fig, ax = plt.subplots(figsize=(16,16))
ax.plot(nclusters, los_val,c='g')
for i, txt in enumerate(np.round(los_val,3)):
    ax.annotate((nclusters[i],np.round(txt,3)), (nclusters[i],los_val[i]))
plt.grid()
plt.title("Loss value for each K")
plt.xlabel("K's")
plt.ylabel("Error measure")
plt.show()
```

In [8]:
```
tfidflabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/tfidf/tfidflabels.pkl')
df_tfidf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tf
idf/df_tfidflblftname.pkl')
optk = np.unique(tfidflabels).shape[0]
```

In [9]:
```
for i in range(optk):
    a= list(df_tfidf[df_tfidf.lbl == i]['ftname'])

    str1 = ' '.join(str(e) for e in a)
    if len(str1) > 0:
        if i == 2 or i == 10:
            plt.figure(figsize=(16,16))
            wordcld = WordCloud(width=1000, height=1000, background_color =
'white',min_font_size=14).generate(str1)
        else:
            plt.figure(figsize=(10,10))
            wordcld = WordCloud(width=500, height=500, background_color =
'white',min_font_size=8).generate(str1)
        plt.imshow(wordcld)
```

square
originally
hot
spend
wheat
watch
soup
spent
style
0
100
200
300
400
0
100
200
300
400

flavor
almond
bread
mean
mix
chip
good
bad
pancake
brands
aid
happen
potato
anywhere
always
best
deep
add
decided
tasting
bonus
black
try
day
definitely
recommend
fried
diet
free
shipping
fruits
gluten
found
german
popcorn
packed
dollar
hydrogenated
expected
nice
medium
premium
say
pounds
night
miss
harmony
preference
nearly
mixes
job
uses
power
overly
honestly
naturally
items
better
us
full
pan
refund
possible
dented
farms
open
years
one
quality
needs
marzano
goes
nut
diagnosed
longer
many
months
0
100
200
300
400

0
200
400
600
800
1000
0
200
400
600
800
1000
craving
love
apples
although
preservatives
amazon
crocker
order
dont
aroma
instructions
disease
anyway
creamer
crisp
agree
caramel
weather
crackers
crunch

0
100
200
300
400
0
100
200
300
400
freeze

thanks.
thicker
reminds
good
times
thin
teas
bit
third
tastes
0
100
200
300
400
0
100
200
300
400

great
good
taste
chocolate
chip
price
find
flavor
time
amazon
love
food
much
coffee
oil
calorie
recommend
tried
best
mix
one
waffle
fat
even
product
free
feel
easy
every
amount
cup
store
rice
many
bitter
favorite
tea
salt
use
eat
dog
better
never
really
different
coconut
enjoy
make
anything
stuff
pop
grocery
hot
dinner
brand
thank
reasonable
thought
syrup
drink
away
old
problem
sweet
wait
difficult
local
thing
customer
tasting
definitely
highly
read
work
go
may
order
hour
mountain
expensive
coat
trans
want
animal
meal
add
picky
per
pod
grove
far
though
butter
vinegar
ago
vegetable
pleasantly
real
quite
less
care
point
day
reviewer
square
cookie
review
egg
excellent
dark
case
regular
hand
enough
provide
package
green
know
think
reasonably
first
baking
disappointed
expiration
olive
next
treat
buy
priced
try
big
newman
looking
pancake
onion
texture
need
packet
earth
salad
contain
low
able
swiss
date
type
right
month
bottle
value
turn
eater

make
iced
ingredient
kettle
jar
chips
kid
level
local
label
last
know
low
light
live
regular
love
lime
coffees
commercial
complained
harder
heavy
hazelnut
product
line
one
drink
idea
hard
listed
taste
large
forward
littlefat
incredible
bisquick
happy
italian
intense
fluffy
calories
machine
bit
lost
worry
item
lb
keeps
lover
hot
major
key
learned
icicle
cream
come
loaded
stores
main
list
im
long
made
house
dust
lived
kick
later
hate
internet
complain
lack
japanese
lemon
late
pie
issue
kit
jack
loose
looked
lamb
let
liquid
loved
introduced
kibble
drinking
grocery
individual
important
brand
jalapeno
healthier
look
instead
lots
sure
mail
time
ice
kona
icing
kernels
compare
tea
leaf
loaf
maker
impossible
cocoa
liked
leave
increase
life
complaint
0
100
200
300
400

sausage
shop
within
cheddar
save
subscribe
cheese
overly
watching
0
100
200
300
400
0
100
200
300
400

cream
glad
fingers
maker
fell
finish
forward
found
coffee
feeding
used
mixed
much
0
100
200
300
400
0
100
200
300
400

whole
sauce
store
food
choice
sugar
ounce
bag
sweetener
product
sour
state
snack
spicy
size
sound
wheat
soy
basically
basil
watchers
absolutely
christmas
options
stay
delicious
way
still
waste
star
strong
clean
brewer
oreo
ordered
based
baby
classic
breed
bacon
others
back
base
clear
basis
basic
cheaper
checked
organic
0
100
200
300
400

crocker
getting
bought
guess
greasy
bottom
pizza
biscuit
beyond
beverage
blood
products
break
gum
guests
betty
guy
beware
boost
sweeter
fan
big
family
asked
guilt
loves
wants
season
nutrition
free
change
0
100
200
300
400

0
100
200
300
400
0
100
200
300
400
product
great
blend
high
business
buttermilk
kept
horses
kitchen
information
dessert
effective
biscotti
absolutely
couple
disgusting
happened
ground
ingredient
feel
liver
chowder
children
ca
flavor
cafe
decided
addition
celiac
give
amazon
called
sauce
gotten
jerky
burnt
cost
girl
hot
benefits
way
intake
love
kind
arrived
brown
flavored
fall
bag
likely
blueberry
worth
date
bunch
taste
boxes
cups
daily
drop
fantastic
considering
quality
calories
store
grown
melted
cinnamon
cacao
lbs
ago
become
compared
least
blue
handy
better
gross
given
cook
cocoa
keeping
definitely
healthy
child
ahead
tasted
control
hard
list
drink
batter
cat
looks
carb
cut
little
bean
cooking
mix
milk
bulk
dry
pet
brand
computer
adding
caffeine
immediately
calcium
consuming
hooked
buying
half
bar
gourmet
time
fit
individually
dish
less
bring
tea
dressing
bite
italy
honey
certainly
brought
likes
messy
companies
delicious
best
price
call
harmony
concerned
huge
common
eat
pretty
choose
available
condition
allowed
convenience
brother
babies
lovers
buds
five
due
cool
corn
except
lot
fiber
left
de
per
one
clam
combination
containers
coming
cake
dairy
pancake
bisquick
cents
black
luck
low
wrapped
lunch
banana
bigger
funny
delivery
comparison
pack
deal
lovely
heart
beat
make
different
broken
apparently
helps
description

say
selling
share
sell
ship
shape
shake
country
seed
seasoning
served
sensitive
service
seem
seasoned
sense
shared
second
serve
serving
set
several
see
setting
send
0
100
200
300
400
0
100
200
300
400

dried
go
finally
buy
back
grains
followed
find
folks
good
feeling
flavor
sample
forever
gold
double
wrong
locally
fish
0
100
200
300
400
0
100
200
300
400

cat
tell
shots
taste
grass
maple
0
100
200
300
400
0
100
200
300
400

works
way
difference
amazon
mrs
friend
well
none
found
0
100
200
300
400
0
100
200
300
400

year
general
freezer
yum
gain
cup
front
flavor
bargain
mic
tangy
gift
number
flour
grams
friends
rice
wrapped
full
one
old
disappointing
hot
grain
rich
frozen
muffins
takes
worth
target
yellow
right
yogurt
yet
chemical
cracker
pleased
try
roast
garbage
nutritional
got
give
frosting
mg
nothing
xlr
taking
yes
packs
free
wrappers
barbeque
say
noticed
gingerbread
write
wow
0
100
200
300
400

satisfied
full
food
mill
hodgson
higher
anymore
years
look
health
hoping
zero
longer
inside
digestive
hot
lunches
month
water
bodied
0
100
200
300
400
0
100
200
300
400

bake
watery
0
100
200
300
400
0
100
200
300
400

oatmeal
diet
fairly
pineapple
fun
pain
science
future
0
100
200
300
400
0
100
200
300
400

junk
iron
able
loves
melt
find
slight
rose
ginger
mint
peppermint
0
100
200
300
400
0
100
200
300
400

vitamin
visit
0
100
200
300
400
0
100
200
300
400

[5.1.5] Applying K-Means Clustering on AVG W2V, SET 3

In [3]:
```
# Please write all the code with proper documentation
```

In [ ]:
```
        aw2v =asign10tsk1()
        aw2v.read_avgw2vec_dt()
        print(aw2v.X_train.shape)
        aw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/av
gw2vec/'
        #tfidf.X_train = tfidf.X_train[:500]
        mink = 2
        maxk= int(math.sqrt(aw2v.X_train.shape[1]/2)+1.0)
        print(maxk)
        aw2v.findkinkmeans(mink, maxk)
        aw2v.avgw2veckmeans()
        del aw2v
```

In [13]:
```
los_val= joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/avgw
2vec/lossval.pkl')
mink= 2
maxk= int(math.sqrt(2000/2)+1.0)
nclusters = [x for x in range(mink,maxk)]
```

In [14]:
```
print(nclusters,len(los_val))
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31] 30
```

In [41]:
```
fig, ax = plt.subplots(figsize=(16,16))
ax.plot(nclusters, los_val,c='g')
#for i, txt in enumerate(np.round(los_val,3)):
#    ax.annotate((nclusters[i],np.round(txt,3)), (nclusters[i],los_val[i]))
plt.grid()
plt.title("Loss value for each K")
plt.xlabel("K's")
plt.ylabel("Error measure")
plt.show()
```

[5.1.6] Wordclouds of clusters obtained after applying k-means on AVG W2V SET 3

In [10]:
```
bowlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/a
vgw2vec/avgw2vlabels.pkl')
bowrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/av
gw2vec/avgw2vwrdcld.pkl')
df = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/avgw2ve
c/df_avgw2vlblftname.pkl')
```

In [11]:
```
print(bowlabels)
print(np.unique(bowlabels))
optk = np.unique(bowlabels).shape[0]
```

```
[13  8  6 ... 16 10  1]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [12]:
```
for i in range(optk):
    a= list(df[df.lbl == i]['ftname'])

    str1 = ' '.join(str(e) for e in a)
    wordcld = WordCloud(width=1000, height=1000, background_color = 'white'
,min_font_size=14).generate(str1)
    plt.figure(figsize=(16,16))
    plt.imshow(wordcld)
```

wabash
turning
keurig
emeril
ale
tully
drinkable
block
period
air
trip
grove
van
slow
considered
0
200
400
600
800
1000

food
know
brand
serve
may
supermarket
absolute
ages
afterwards
accidentally
advantage
anticipated
accident
addition
anyways
appeared
break
apart
biting
bisquick
bad
badly
address
airtight
active
aside
asian
authentic
bananas
cake
choco
closed
coating
contacted
breast
bisque
boiling
chilled
class
attempt
category
children
caused
cholesterol
cook
bag
cure
clam
almond
american
audacity
bottle
clean
bar
areas
blue
crumble
benefits
case
beyond
bland
cheesecake
cigars
containers
cucumber
counting
college
brewer
community
crumbly
craving
digestion
deserves
doubted
changed
closest
cinnamon
books
complex
creamer
drinks
drank
credit
currently
diet
diarrhea
dishes
damage
decaf
darker
dirty
doubt
difficulty
difference
dressings
delivers
drivers
dump
door
eating
finding
grateful
common
chili
icicle
francisco
hibiscus
diagnosed
failure
dense
fork
frozen
gassy
damaged
everything
give
feels
easy
eat
drink
kona
herb
feed
haha
keeping
likes
japanese
directions
hamburger
expiration
gum
guests
hotter
finish
hell
freshness
cut
kroger
freezer
exceptionlabeled
inside
lime
high
grind
flaky
goal
juicy
dont
gone
falling
hines
labels
happened
heard
fast
dane
layer
decent
glossy
ladies
february
hold
fishy
experiencing
fries
favorite
fluffy
energy
fan
example
exciting
come
desired
0
200
400
600
800
1000

calling
cracked
checking
bank
amazing
recomend
real
named
remain
pumpkin
fix
jalapeno
dish
stupid
complete
exactly
sharp
wrapper
slightly
whatever
us
prime
preferred
replacement
experience
crisp
appeal
cheeses
europe
house
oreo
prevent
dip
boost
yum
snack
complained
pastas
seattle
minus
schedule
highest
portion
leaving
loads
fridge
somewhere
letting
remaining
prices
wife
round
instantly
fanatic
runner
tips
mother
fortunes
reviewers
mills
hurt
thus
consider
asked
show
friend
places
yr
disappointment
grill
transfer
bummed
wafer
throat
pull
personally
drinker
target
neat
liquor
poultry
sent
several
avoid
proteins
pure
sticks
volume
went
party
jam
durable
berries
ball
delight
syrup
receive
sandwich
fuss
bonus
title
expected
stayed
department
hand
remind
spend
aware
cause
avoiding
helped
tomatoes
colorful
directed
vendors
favorites
treat
full
agreed
0
200
400
600
800
1000

grilled
brittle
protein
missed
cents
clumped
heaven
acidity
website
mark
unsalted
snacking
really
tin
mountain
hoped
bits
evenings
intense
bags
complaining
interesting
lists
able
flowers
partially
reorder
positive
reliable
test
worse
button
vacuum
success
wondered
bowl
nutrients
funny
sooo
workout
iraq
ara
total
trick
ended
carrots
sisters
excited
0
200
400
600
800
1000

advertised
headache
start
absolutely
compare
alternatives
background
belgian
burning
aluminum
asparagus
baxter
baby
card
bears
brownies
burnt
buttermilk
bolder
brought
beneful
beware
caffeine
buck
cabinet
basis
carb
charged
compatible
coast
fruit
carefully
fudge
finely
draw
factor
hodgson
glycemic
individual
glucose
formulas
human
magic
email
controlled
kibble
follows
cookbook
hours
regret
mrs
lost
scary
hooked
fun
matter
crappy
lemons
folger
pressure
innova
homemade
initial
gobbled
magazine
distinctive
excessive
creates
commented
checked
delicious
caviar
natural
simple
melitta
farm
hope
drawback
canned
lemonade
happens
january
essential
device
passed
junk
cal
looking
dipping
definite
dressing
discount
ice
cranberry
inches
pounds
signed
play
mushy
mcdougall
cutting
leche
money
constantly
justify
kirkland
quinoa
mix
multi
chew
fixed
list
pieces
hulless
lunches
covered
italian
paired
mono
petco
pet
delighted
duty
hearty
honest
con
sale
spots
pancakes
kcups
reviewing
single
pouch
standing
muffins
researched
salts
driving
eater
fillers
jack
club
major
economical
recently
lays
early
medium
sticking
meets
roughly
movements
rare
rainforest
riesling
needs
reasonably
shrimp
movie
cupboard
factory
number
disappointed
produced
problem
progresso
plockys
rye
overpower
perhaps
pitcher
mmm
overweight
online
said
saved
safeway
ride
rid
neck
reminiscent
skinny
satisfied
paste
seasoned
might
milk
hazelnut
pleasantly
pocky
remembered
sound
sick
pan
rolled
0
200
400
600
800
1000

gotten
kid
sending
kick
friends
onto
improvement
situation
trusted
daughter
range
firm
herbs
listed
packaged
0
200
400
600
800
1000

advanced
accessible
dissolve
adopted
acid
advance
agree
allergies
balanced
basilarizona
benefit
biggest
blood
combined
congestion
cajun
around
claims
consumed
cups
discontinued
bed
brown
boxes
call
acts
among
alpine
busy
burned
cart
course
chance
cheddar
choice
catch
bowls
big
carbs
cats
curry
concentrated
cooked
contains
convinced
brownie
crocker
country
designed
custard
cross
couldnt
dash
deceptive
downside
disgusted
eaten
hunting
exotic
olives
hummus
instead
knife
lovers
done
items
packets
pretzel
riviera
pizzas
described
evening
depending
european
magnesium
quantities
soft
obvious
says
one
gold
parts
stash
softer
soy
sacrifice
replacing
quite
opening
surprise
insanity
pb
producing
secure-y
stevia
gave
fragrance
pots
pricy
mine
rest
fills
pocket
pairing
removed
super
served
subjective
shame
sweetened
edible
looked
lives
eye
limited
grocery
switching
palm
fattening
guayaki
four
dash
habit
prickly
microwave
method
onions
french
happen
plate
leaf
lo
peanuts
rounded
match
purchase
muesli
needed
ruin
shepherd
jelly
smaller
grass
organic
introduced
marzano
filtered
minced
soooooo
marketed
saw
sizes
die
sometimes
sweet
eve
produce
repeat
matters
tasteless
splenda
mom
parent
potato
stand
hurry
guess
sprouts
florida
marinade
earlier
offered
everywhere
dying
member
recommended
scent
spoon
morning
recent
raisin
smoothness
peppers
flaxseeds
residue
sand
promise
refer
pretty
ounce
suggested
memories
ribs
side
java
spring
numerous
0
200
400
600
800
1000

consistency
creamers
environment
previously
vegetarian
appropriate
sounds
hair
began
forget
following
boneless
seven
fats
gas
paleo
electrolytes
shipping
garlic
whirley
ear
airport
vinegar
donut
gain
leaves
babka
baseball
mile
calories
time
coffees
amazed
men
theme
vitacoco
sign
stale
vodka
bull
dissolved
desert
filter
peanut
strong
inhaled
slight
mocha
affordable
handling
tictac
wondering
revolution
favorable
eggnog
assume
gevalia
sauces
dove
cocoas
constipation
check
wellness
voted
0
200
400
600
800
1000

newman
deal
grow
result
turn
account
addict
adore
americans
antioxidants
alternative
argentina
aftertaste
anybody
almost
asking
balance
called
average
barley
candies
beverage
cornstarch
counter
consume
caffeinated
cleaning
devours
continued
available
container
breaks
crisps
cool
cute
comparable
delectable
classic
date
cases
died
donated
generous
gradually
followed
fortune
godsend
dogs
donuts
ends
decaffeinated
hoping
enfamil
fatty
happier
colic
charity
bill
boys
brush
carbon
decades
base
consumption
bulk
holistic
balls
coffee
finicky
bones
fans
friendly
creative
fussy
asia
costs
care
cereal
evo
days
fault
falls
gets
cafe
breading
environmental
coconuts
exceptional
behind
fry
compromise
costco
audio
audible
batter
cayenne
al
apples
connoisseur
healthier
eager
chocolately
anything
deliver
delivered
brew
bisquik
fiber
hint
cento
general
advertising
faster
feedback
famous
cable
coffe
gourmet
darn
flavors
bagged
caramels
cupcakes
decision
herbal
bold
cold
even
heaping
easily
ground
castor
finishing
coloring
carbohydrate
forward
changes
afford
bite
candy
forgot
entirely
bowel
color
crash
flax
affect
altoids
dissapointed
asks
assortment
artifical
automatically
crab
calls
carries
couple
bread
goo
comparing
feeding
grains
grape
grandson
fornula
bitter
cubes
beginning
coming
cover
finds
go
corn
fake
figured
cant
freeze
best
broke
decaff
height
create
hesitate
filters
eyes
encourage
hands
grainy
hey
fennel
hit
driver
hershey
lakes
frosting
drunk
0
200
400
600
800
1000

type
chicken
acceptable
addictive
allergic
amount
bodied
ahead
advice
agave
age
ago
america
aspect
auto
anymore
animal
bitterness
breakfast
blender
business
boot
beverages
bacteria
becomes
chocolate
chemical
chronic
caribou
cans
compact
celestial
colada
chews
brewed
change
cheaper
britt
bother
choke
crowd
dogfood
dates
deep
correctly
chopped
colorings
drinks
buds
easier
flaxseed
kombucha
compliment
cooking
deserve
paid
metal
grocers
melts
fig
eats
lines
oil
nowhere
decided
fortunately
desk
degrees
future
frying
parsley
mailed
lbs
convenient
looks
lacking
middle
ships
iced
glass
proven
neither
popcorn
nustevia
electric
expert
holidays
shower
send
quit
new
owner
fine
nasal
favor
detect
mention
generally
lots
kavli
fluoride
definitely
overs
clearly
channel
floor
raw
healthy
dead
face
proud
dollars
parties
comment
manufacturer
experimented
maltodextrin
longer
granted
escapes
guilt
plain
nine
members
dieting
share
felidae
lining
meet
goat
glaze
lock
hotel
hate
offers
powering
sachets
safe
much
pina
organics
included
main
options
machine
process
profits
current
expired
question
internet
mean
perfectly
kind
mail
routine
bigger
inexpensive
leavening
handful
learn
plant
getting
ill
reduce
post
missing
pastry
pre
refused
rather
plugged
ensure
ca
potassium
personal
mass
right
lemon
mart
lets
mints
indoor
requires
inedible
direct
sample
shake
powerful
lately
follow
loaded
irish
liked
pre

adding
take
actually
allows
although
adjust
additive
aromatic
ai
arrived
arabica
april
attractive
born
believe
bills
beats
batteries
bean
canola
breath
careful
bought
blends
boxed
broth
clams
concern
coupon
chalk
cat
charging
confirmed
consistently
comparisons
commercial
buying
crackers
diamond
discovered
crushed
crispy
continue
dandruff
given
connect
folks
fresh
first
flavour
fall
favors
girlfriend
city
conventional
experienced
elegant
family
dumplings
customer
prescription
great
enjoy
heinz
dispenser
hungry
disorder
ingredient
learning
neighbor
smells
ordinary
thanks
rescue
pressed
gained
manufactured
recieved
definately
frequently
hill
design
spice
markets
remember
jasmine
people
north
singles
smoothies
suggestion
sure
crunchy
steamed
pass
rave
profit
pizza
stretch
realized
minerals
noted
sites
ran
marks
person
report
facts
diced
flavor
cream
intact
lacks
remains
health
pate
dha
roses
ratings
pretzels
messy
pour
squares
teabags
pops
import
stakespanel
teeth
fda
spiced
system
style
mac
mild
mixing
salt
immediately
photo
equipment
roasts
older
opposed
review
occasion
make
minor
sorghum
skillet
lighter
medicinal
favors
sons
sliced
kettle
loyal
lol
pouches
shared
iron
fence
mackerel
season
filler
kept
mushrooms
rotate
practices
petsmart
organix
poor
substantial
dumplings
superior
playing
nuts
indian
little
lobster
kinds
hence
nobody
recommends
lasted
tang
pale
reading
last
shared
0
200
400
600
800
1000

qualities
chewy
hates
thins
spread
image
indulgence
locate
spearmint
carried
watchers
packet
pc
celebrate
halloween
hopefully
apricot
prefer
tv
ray
everybody
poured
microphone
understood
learned
carton
heavily
particular
chinese
head
0
200
400
600
800
1000

reaction
additives
acquired
additional
alcohol
alteracai
assorted
arriving
anti
across
attached
back
bring
ate
aunt
babies
barbecue
barbeque
calcium
baked
became
apple
awhile
bliss
brainer
cleaned
collection
contain
close
jamaican
inclined
increased
dioxide
dessert
celiac
depends
conscious
manufacturers
betty
buyer
caveat
batches
causes
consumers
footprint
cookie
concerned
cuisinart
box
buster
brother
cane
crumbled
costly
certaincakesters
encountered
grandchildren
filled
effects
eventually
meal
office
despite
index
kids
grinder
night
pamela
insulin
includes
nothing
living
okay
hear
normally
notice
miles
knew
diabetes
cooler
ears
group
inch
hunger
india
live
ideal
character
changing
chemicals
constant
consistent
competitive
diabetic
microphones
compleats
graham
earl
father
granola
doggie
dropped
cough
fingers
licked
king
hide
ny
el
omaha
message
kernel
paprika
experiment
ham
nose
harder
marked
loud
combine
drip
levels
international
imo
comes
growing
fancy
jello
heavier
mostly
everyday
dark
helpful
incredible
medication
moderate
guys
olive
naturals
lactose
event
honey
guest
kosher
disgusting
judging
due
five
jars
gives
hour
many
lentil
large
lard
meats
nut
parents
def
0
200
400
600
800
1000

ahmad
add
min
allowed
amounts
amish
always
animals
another
answer
butter
desserts
charming
cheap
barely
flavorful
baking
buffalo
biscotti
artificial
brain
appealing
aspartame
automatic
chocolaty
baker
cereals
chose
difficult
coupons
central
correct
dinners
cilantro
condiment
bully
camping
belly
chalky
brews
companies
beach
budget
bright
charge
diaper
bake
beer
causing
condition
batch
chip
busted
based
bathroom
broken
california
farmhouse
fabulous
duncan
dog
help
found
extract
feet
front
empty
finger
figure
neighbors
holds
dust
infection
drinking
caring
expect
hardware
dipped
drive
charger
holes
kernels
mate
lab
green
merchandise
offer
maybe
frig
improve
oddly
local
hardly
melt
oils
dough
habits
love
mango
ones
meaning
hormones
ener
half
enjoying
genetically
felt
noodles
nachos
home
jerk
load
ole
grandmother
filling
freshly
lasting
minute
gonna
kit
except
google
info
mixes
everyone
flours
litter
equally
mellow
mid
co
creme
hesitant
dijon
moving
heavy
dietary
greasy
metallic
lend
disappoint
improved
goes
million
jamaica
length
crust
express
finally
machines
nutritious
item
error
disappointing
moved
meal
childhood
flimsy
hated
guarantee
month
monthly
manage
later
measuring
lips
exclusively
comforting
expecting
mislcading
maintain
football
occasions
mill
created
flavoring
grown
go
opened
mistake
information
gifts
rutro
complaint
kitty
matcha
grade
dislike
nice
confused
dairy
met
fantastic
oak
gross
laptop
multiple
0
200
400
600
800
1000

cocker
differently
enjoyed
creamier
require
area
backyard
view
small
ingredients
thirst
lying
harbor
burn
blow
bites
beautifully
boyfriend
cocoa
cracker
breed
processed
mixer
luck
vocals
safflower
er
tray
acids
delivery
reducing
opportunity
cheapest
coat
concerns
rides
gooey
sensitive
significant
lightweight
substitute
loaf
max
switch
shipment
pulled
waited
double
hips
software
nutritional
ketchup
groceries
select
refined
bars
already
zip
mug
till
contained
berry
flare
martini
seconds
interest
trying
bacon
leftover
pricey
came
holding
gf
merrick
melted
vitamin
china
breaking
afraid
near
recall
de
dozen
thyme
occasional
reordering
creaminess
gotta
sushi
computer
mercury
sales
efficient
coca
enough
choose
saturated
turkey
roland
leave
attention
severe
surely
job
typically
powder
soap
teabag
increase
skin
loved
note
infections
remained
immediate

soup
ability
serving
twist
girl
accustomed
addicted
ad
enhance
almonds
bucks
arrives
associated
cravings
caution
citrus
anywhere
bloody
book
bunch
choices
beef
carrot
ask
aisle
blame
alot
buffet
claim
cider
bob
celery
along
baronet
begin
blendblueberries
band
assuming
banana
black
canister
cheese
bottom
appreciate
becoming
chamomile
broccoli
blah
admit
canadian
avid
adds
bottles
beans
birthday
cancelled
apparently
cashew
bugs
adult
child
bargain
boy
bang
appears
chef
additionally
aid
cherry
comfort
caf
canidae
besides
bringing
away
chunks
discover
curious
crumbs
equivalent
dial
final
hundreds
0
200
400
600
800
1000

biscuit
added
recipe
year
absorb
adults
ahoy
alive
afternoon
allergy
aged
allowing
antibiotics
amazingly
alone
amazon
awful
anyway
appear
celiacs
challenge
cashews
certified
chips
charges
better
chowder
antioxidant
bbq
body
burgers
boil
boring
bubble
breads
buy
buttery
colors
chunk
carry
coco
caramel
coats
cacao
carrying
bees
combination
complaints
control
cornucopia
cornbread
complexity
concentrate
condenser
creating
component
chocolates
compromised
combo
carrying
drops
duck
doctor
easter
expensive
employees
feature
fashioned
grew
describe
gravy
gluten
hard
eaters
handle
harmful
hormel
highly
giant
leftovers
liquids
industry
intolerant
forced
provide
minimum
maltitol
imported
itching
poison
pomegranate
mildly
gritty
preamp
mood
explain
issues
known
marshmallows
owners
micro
gmo
mini
pups
rating
rip
ridiculously
ridiculous
reduced
pros
put
notch
production
material
loving
licorice
phosphate
infant
rain
pop
find
quench
rolled
recognize
maximum
mics
latte
crave
fewer
ripped
packed
milder
diets
negative
god
promised
critical
guessed
fair
husky
knob
rough
garage
dried
germany
kidding
depth
lesser
fully
frankly
healthiest
luzianne
meant
opted
omg
completely
listing
heavenly
roll
purpose
liking
points
doesnt
cornbread
customers
malt
edit
likely
concept
drinkers
late
overall
run
glad
especially
considering
gummy
fit
needless
nabisco
law
puck
enjoyable
gary
insane
em
hits
poorly
move
grab
oats
dollar
moral
raised
pictures
present
digest
rcognized
page
0
200
400
600
800
1000

hulls
settled
taffy
beautiful
flavored
studio
minty
kool
basic
erin
forever
distinct
df
fruits
bluegrass
sensitivity
cocktail
look
gatorade
analysis
puff
sumatra
tassimo
roaster
gm
canin
daughters
mahogany
greatest
popular
addicting
ups
thai
guy
cardboard
latest
tahini
0
200
400
600
800
1000

bakery
acid
flow
acid
airy
aroma
basket
bet
brings
canada
anyone
blueberry
basically
become
calorie
chocolatey
certainly
chai
costa
couscous
cookies
chain
decafs
confection
consuming
every
extreme
determine
happily
contents
convert
earth
instant
history
essentials
effort
edge
giving
grease
garden
goodies
comparison
according
bpa
eight
idea
holiday
glue
goods
grapefruit
heart
lazy
gums
ginger
halal
exact
limit
foodie
market
grey
flat
kill
egg
cooks
etc
superb
flavorless
reviewer
surprised
suspicious
similac
sausage
sandwiches
farmers
refrigerator
fail
yuban
pungent
skeptical
solution
usb
packaging
id
persons
far
extremely
overly
receipt
senior
kitties
mayonnaise
mind
soooo
ready
none
shapes
include
things
rate
wins
staff
trips
spectrum
supplement
sips
smokey
richness
skins
tend
satay
option
stays
root
pup
yellow
grain
treatment
marketing
pg
original
occasionally
well
sucralose
preference
pets
mess
realize
soil
refreshing
safety
waiting
white
special
reviewed
within
various
space
whether
quest
road
vendor
prior
window
stopped
someone
words
versions
thoughts
wings
flower
superb
0
200
400
600
800
1000

[5.1.7] Applying K-Means Clustering on TFIDF W2V, SET 4

In [3]:
```
# Please write all the code with proper documentation
```

In [ ]:
```
        wtw2v =asign10tsk1()
        wtw2v.read_tfidfwtw2vec_dt()
        #print(wtw2v.X_train.shape)
        wtw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
fidfwtw2vec/'

        mink = 2
        maxk= int(math.sqrt(len(wtw2v.X_train)/2)+1.0)
        print(maxk)
        wtw2v.findkinkmeans(mink, maxk)
        wtw2v.tfidfwtw2veckmeans()
```

In [57]:
```
los_val= joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tfid
fwtw2vec/lossval.pkl')
mink= 2
maxk= 50  #int(math.sqrt(50/2)+1.0)
nclusters = [x for x in range(mink,maxk)]
```

In [58]:
```
print(nclusters,len(los_val))
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49] 48
```

In [60]:
```
fig, ax = plt.subplots(figsize=(16,16))
ax.plot(nclusters, los_val,c='g')
#for i, txt in enumerate(np.round(los_val,3)):
#    ax.annotate((nclusters[i],np.round(txt,3)), (nclusters[i],los_val[i]))
plt.grid()
plt.title("Loss value for each K")
plt.xlabel("K's")
plt.ylabel("Error measure")
plt.xticks(np.arange(0, 51))
plt.show()
```

[5.1.8] Wordclouds of clusters obtained after applying k-means on TFIDF W2V SET 4

In [3]:
```
# Please write all the code with proper documentation
```

In [10]:
```
bowlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/t
fidfwtw2vec/wtw2vlabels.pkl')
bowrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tf
idfwtw2vec/wtw2vwrdcld.pkl')
df = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tfidfwtw
2vec/df_wtw2vlblftname.pkl')
```

In [15]:
```
for i in range(6):
    a= list(df[df.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)
    #print(i,str1)

    if i in (1,2,3,4) :
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=14).generate(str1)
        plt.figure(figsize=(16,16))
    else:
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=8).generate(str1)
        plt.figure(figsize=(10,10))
    plt.imshow(wordcld)
```

emeril
drinkable
van
ale
wabash
considered
keurig
period
trip
air
slow
tully
grove
block
turning
0
100
200
300
400
0
100
200
300
400

know
may
serve
brand
food
supermarket
accidentally
absolute
chilled
badly
ages
advantage
cholesterol
difficulty
feels
creamer
darker
delivers
hines
hold
contacted
audacity
counting
apart
community
dont
brewer
books
difference
diagnosed
changed
children
bar
currently
flavorings
hamburger
experiencing
crystals
dishes
gone
doubled
airtight
address
deserves
finish
favorite
fell
aside
come
grateful
grinding
dane
exciting
feed
bag
clean
break
grams
cure
delicate
hole
doubt
accident
crumbly
icicle
damaged
dressings
anyways
chili
benefits
beyond
addition
class
fresher
example
heard
easy
fork
containers
fries
decent
bad
energy
expiration
category
guests
bananas
damage
areas
biting
freshness
fool
flaky
bisquick
college
case
falling
boiling
coating
francisco
desired
different
failure
dense
complex
bisque
goal
hotter
garbage
credit
anticipated
exception
cucumber
glossy
asian
common
bottle
cut
breast
everything
cheesecake
frozen
day
fluffy
hum
directions
attempt
gassy
hibiscus
diet
decaf
drivers
choco
cigars
closest
gum
digestive
herb
cook
finding
caused
door
dump
february
clam
give
happened
american
active
blue
fav
cinnamon
eating
diarrhea
appeared
grind
high
bland
freezer
eat
cake
craving
fishy
beat
afterwards
fast
authentic
almond
drank
haha
crumble
closed

portion
wafer
amazing
agreed
build
bummed
berries
ball
boost
avoid
avoiding
aware
asked
appeal
bonus
calling
checking
cheeses
cause
bank
complained
drinker
crisp
consider
department
delight
favorites
expected
grill
friend
fridge
europe
helped
cracked
directed
fanatic
exactly
fix
party
durable
fortunes
full
complete
colorful
disappointment
fuss
dip
pull
liquor
remain
experience
highest
prevent
prices
round
pure
poultry
thus
seattle
several
leaving
spend
somewhere
transfer
slightly
vendors
prime
sandwich
treat
runnier
remind
throat
hand
preferred
jalapeno
proteins
remaining
real
whatever
reviewers
neat
sticks
went
jam
tips
minus
mother
sent
recomend
snack
replacement
letting
pastas
instantly
title
schedule
hurt
volume
pumpkin
dish
wrapper
loads
mills
tomatoes
stayed
receive
sharp
oreo
stupid
target
show
house
named
wife
yum
syrup
us
personally
0
200
400
600
800
1000

test
lists
wondered
mountain
button
bowl
sooo
flowers
intense
carrots
workout
ara
excited
sisters
grilled
total
unsalted
partially
bits
reorder
iraq
nutrients
really
website
snacking
ended
clumped
positive
bags
missed
mark
vacuum
funny
reliable
cents
heaven
interesting
brittle
success
trick
protein
acidity
tin
hoped
evenings
complaining
worse
0
200
400
600
800
1000

alternatives
compare
aluminum
asparagus
background
headache
bears
brought
advertised
cabinet
carefully
brownies
cal
start
burning
beware
bolder
canned
charged
baby
basis
card
buck
burnt
cloves
carb
buttermilk
checked
absolutely
baxter
belgian
beneful
caffeine
excessive
honest
controlled
delicious
economical
eater
distinctive
hearty
crappy
initial
lays
hulless
coast
cookbook
cranberry
creates
gobbled
commented
brownies
glucose
email
fillers
happens
finely
italian
lemonade
kcups
cabinet
carefully
folger
major
fixed
melitta
signed
list
cupboard
hooked
milk
medium
driving
sticking
online
caviar
follows
pieces
kibble
formulas
early
farm
innova
device
hours
card
buck
might
human
cloves
junk
hope
mushy
seasoned
delighted
mono
checked
definite
cutting
skinny
set
carb
buttermilk
club
looking
kirkland
reasonably
produced
money
essential
drawback
dipping
overpower
mcdougall
meets
constantly
fruit
perhaps
fun
inches
neck
mix
beneful
belgian
glycemic
fudge
0
200
400
600
800
1000

[5.2] Agglomerative Clustering

[5.2.1] Applying Agglomerative Clustering on AVG W2V, SET 3

In [ ]:
```
import numpy as np
import pandas as pd
import pickle
import math

from datetime import datetime
from sklearn.cluster import AgglomerativeClustering
from sklearn.externals import joblib

class asign10tsk2 :

        def __init__(self):
                self.X_train =pd.DataFrame()
                self.ipfname = ''
                self.opfname = ''


                self.nclusters = []

                self.labels = None

                self.w2vecfeat = []
                self.w2vecwrdcld = []

                self.tfidfeat = []
                self.tfidfwrdcld = []

        def read_avgw2vec_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/avgw2vectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'w2vec_feat', 'rb') as fp:
                        self.w2vecfeat = pickle.load(fp)

        def read_tfidfwtw2vec_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/tfidfwtw2vectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'tfidf_feat', 'rb') as fp:
                        self.tfidfeat = pickle.load(fp)

        def aggclst_avgw2vec(self):
                self.nclusters = [2,5,7]
                for i in self.nclusters:
                        self.aggclst = AgglomerativeClustering(n_clusters=i
)
                        self.aggclst.fit(self.X_train)
                        self.labels = self.aggclst.labels_

                        avgw2vlblftnme = sorted(zip(self.labels,self.w2vecf
```

```
eat))

                        self.w2vecwrdcld = [word for word in avgw2vlblftnme
]
                        df = pd.DataFrame(self.w2vecwrdcld, columns =['lbl'
, 'ftname'])
                        joblib.dump(self.labels,self.opfname+str(i)+'_avgw2
vlabels.pkl')
                        joblib.dump(self.w2vecwrdcld,self.opfname+str(i)+'_
avgw2vwrdcld.pkl')
                        joblib.dump(df,self.opfname+str(i)+'_df_avgw2vlblft
name.pkl')

        def aggclst_tfidfwtw2vec(self):
                self.nclusters = [2,5,7]
                for i in self.nclusters:
                        self.aggclst = AgglomerativeClustering(n_clusters=i
)
                        self.aggclst.fit(self.X_train)
                        self.labels = self.aggclst.labels_

                        wtw2vlblftnme = sorted(zip(self.labels,self.tfidfea
t))

                        self.tfidfwrdcld = [word for word in wtw2vlblftnme]
                        df = pd.DataFrame(self.tfidfwrdcld, columns =['lbl'
, 'ftname'])
                        joblib.dump(self.labels,self.opfname+str(i)+'_wtw2v
labels.pkl')
                        joblib.dump(self.tfidfwrdcld,self.opfname+str(i)+'_
wtw2vwrdcld.pkl')
                        joblib.dump(df,self.opfname+str(i)+'_df_wtw2vlblftn
ame.pkl')

if __name__ == "__main__":


        aw2v =asign10tsk2()
        aw2v.read_avgw2vec_dt()
        print(aw2v.X_train.shape)
        aw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/ta
sk-2/avgw2vec/'

        aw2v.aggclst_avgw2vec()


        wtw2v =asign10tsk2()
        wtw2v.read_tfidfwtw2vec_dt()
        #print(wtw2v.X_train.shape)
        wtw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
ask-2/wtw2vec/'

        wtw2v.aggclst_tfidfwtw2vec()
```

In [3]:

```
# Please write all the code with proper documentation
```

In [ ]:
```
        aw2v =asign10tsk2()
        aw2v.read_avgw2vec_dt()
        print(aw2v.X_train.shape)
        aw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/ta
sk-2/avgw2vec/'

        aw2v.aggclst_avgw2vec()
```

[5.2.2] Wordclouds of clusters obtained after applying Agglomerative Clustering on AVG W2V SET 3

In [3]:
```
# Please write all the code with proper documentation
```

Number of Clusters 2

In [38]:
```
aw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-2/avgw2vec/2_avgw2vlabels.pkl')
aw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-2/avgw2vec/2_avgw2vwrdcld.pkl')
aw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/task
-2/avgw2vec/2_df_avgw2vlblftname.pkl')
optk = np.unique(aw2vlabels).shape[0]
```

In [40]:
```
for i in range(optk):
    a= list(aw2vdf[aw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)

    wordcld = WordCloud(width=1000, height=1000, background_color = 'white'
,min_font_size=12).generate(str1)
    plt.figure(figsize=(18,18))
    plt.imshow(wordcld)
```

add
amount
american
animal
beverage
bag
container
cereal
bowl
bill
brew
case
beat
anyway
chicken
change
break
drink
driver
juice
flaxseed
evening
date
hold
lid
favor
habit
dissolve
coffee
clam
coupon
deliver
fan
food
herb
fact
fall
deserve
member
raisin
lb
item
lover
product
square
sister
noodle
side
eye
may
son
eat
pizza
keep
neighbor
flavor
deal
hole
pain
dog
min
stand
offer
review
brand
benefit
call
cat
gram
dollar
color
coloring
complaint
issue
mark
kind
know
month
machine
minute
day
give
spice
recording
wine
smoothie
want
use
thank
tablespoon
visit
antioxidants
alpine
adore
almond
recommend
waffle
unit
airtight
accidentally
additionally
profit
pretzel
airport
melt
acceptable
supermarket
plant
acidity
allergic
accessible
anticipated
allows
america
ago
advertising
allowed
addictive
addict
ton
agree
among
adding
agave
advantage
additive
pod
roast
oil
tart
account
ability
aftertaste
ahead
adopted
adjust
ages
ahmad
new
square
store
acts
advice
pie
reward
advanced
absolute
although
smell
another
affect
ai
afford
al
actually
part
always
active
aged
toy
sometime
almost
stop
take
morning
address
turn
acid
altoids
think
result
pepper
wonder
type
organic
grow
go
advance
onion
accident
peanut
allergies
sugar
standard
line
report
flavoring
able
pea
one
lot
reward
alternative
addition
newman
occasion
package
amish
afterwards
ounce
size

Number of Clusters 5

In [8]:
```
aw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-2/avgw2vec/5_avgw2vlabels.pkl')
aw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-2/avgw2vec/5_avgw2vwrdcld.pkl')
aw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/task
-2/avgw2vec/5_df_avgw2vlblftname.pkl')
optk = np.unique(aw2vlabels).shape[0]
```

In [37]:
```
for i in range(optk):
    a= list(aw2vdf[aw2vdf.lbl == i]['ftname'])

    str1 = ' '.join(str(e) for e in a)
    if i in [2,5]:
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=8).generate(str1)
        plt.figure(figsize=(12,12))
    else:
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=12).generate(str1)
        plt.figure(figsize=(18,18))
    plt.imshow(wordcld)
```

adult
cashew
biscuit
buck
coat
consumer
celiac
cause
enhance
point
powder
soup
afraid
absorbafternoon
alcohol
absolutely
awful
awhile
caviar
bitterness
bring
basis
calling
ask
blow
avoiding
appears
burgers
anybody
boil
buy
baronet
boring
away
bottles
aware
breads
bbq
buttery
additional
acids
canidae
canadian
begin
breed
additives
addicted
bob
appeal
boy
aunt
canister
associated
brother
burnt
cacao
appreciate
apparently
twist
drop
banana
alone
arrives
challenge
across
hour
human
anti
barbeque
second
fit
baxter
berry
bank
book
shot
burn
batches
accustomed
purpose
bottom
pancake
amazingly
allowing
birthday
allergy
almonds
alive
arriving
blah
ate
attached
came
run
tip
ear
natural
serving
blueberries
buyer
brownies
avid
broccoli
calcium
brewing
cane
backyard
leftover
problem
level
treat
carry
became
carrying
betty
girl
already
asparagus
certain
celery
tray
anywhere
aluminum
bees
ball
bloody
babies
buffet
cakesters
amazon
caffeine
bringing
assuming
age
apple
brainer
added
coconut
acquired
pound
better
bubble
barbecue
cabinet
ahoy
ride
becoming
seed
along
blame
alter
bacon
antibiotics
admit
asia
caramel
besides
avoid
bliss
beans
ad
berries
asked
antioxidant
chamomile
attention
year
aisle
band
breaking
buster
bang
bummed
spot
aid
acai
baked
back
beef
box
cancelled
body
bugs
alot
agreed
assorted
blend
wafer
salsa
bites
taste
reaction
remind
bars
canned
carefully
seem
baby
0
200
400
600
800
1000

reviewer
content
addicting
meal
vendor
according
advertised
portion
form
acidic
allergens
alternatives
start
piece
allow
airy
anyone
actual
amazing
basically
approximately
basic
beautiful
beautifully
awesome
bakery
carton
carb
anytime
bonus
bears
certainly
become
cocker
bolder
bet
apricot
basket
background
brings
beneful
area
canin
canada
blueberry
bottled
build
bee
celebrate
aroma
carried
bpa
chewing
checked
chocolatey
compare
experience
exact
glucose
department
groceries
edge
essentials
determine
farms
comments
ginger
elsewhere
greatly
gooey
burning
boyfriend
flavorless
cookbook
construction
disappointed
extremely
consuming
consider
crazy
golf
egg
folger
chinese
confection
delight
english
costa
every
early
cheapest
dipping
calorie
creaminess
far
duty
car
eight
chain
extreme
covered
chewy
dented
gatorade
giving
fortunes
delightful
comparison
flour
checking
brought
familiar
couscous
cookies
death
fail
grease
boost
cheeses
fairly
flower
follows
grey
delicious
forever
everybody
fat
complained
constantly
daily
full
crappy
fight
formulas
foil
containing
controlled
drip
cardboard
buttermilk
arrive
china
etc
grain
expected
farmers
bit
club
center
dr
beware
belgian
effort
effect
friend
gums
colorful
coated
card
economy
dip
foodie
charged
choose
free
equal
grill
crap
goods
ghiradelli
gift
glue
goodness
end
grapefruit
habanero
flat
bone
cakes
flavored
cooks
garden
game
dressing
convert
entire
fillers
drinker
decafs
eater
goodies
europe
chai
draw
halal
cal
earth
convenience
glycemic
amazing
actual
awesome
bakery
cardboard
controlled
0
200
400
600
800
1000

previously
bunch
dove
mocha
cocoas
gas
tictac
settled
ale
df
scary
kool
trip
keurig
turning
tully
distinct
voted
hulls
guy
slow
greatest
handling
block
strong
cocktail
daughters
erin
look
amazed
check
vitacoco
drinkable
popular
gevalia
grove
inhaled
seven
ups
latest
wabash
affordable
time
van
whirley
thai
considered
assume
paleo
sumatra
vinegar
vegetarian
tahini
bull
bluegrass
puff
theme
gm
donut
emeril
following
taffy
dioxide
analysis
stale
filter
men
environment
period
sign
minty
creamers
gain
consistency
studio
roaster
wellness
0
100
200
300
400
0
100
200
300
400

dissolve
bowl
acid
additive
accessible
evening
one
advance
alpine
california
acidity
advanced
among
around
bits
brown
cart
bags
ara
bed
cajun
carrots
arizona
button
boxes
brownie
burned
call
cents
chalky
congestion
chance
cheddar
brittle
agree
able
attractive
benefit
complaining
cats
busy
acts
adopted
balanced
contains
country
catch
amounts
carbs
cups
custard
allergies
blood
big
described
consumed
concentrated
claims
clumped
crocker
cons
curry
ended
dijon
grass
grilled
exotic
hoped
diced
die
cooked
packets
everywhere
fragrance
filtered
grandmother
pretzel
edible
earlier
mountain
missed
organic
pricy
java
member
magnesium
muesli
iraq
olives
done
memories
parent
pea
matters
marinade
designed
flaxseeds
instead
leaf
mark
peanuts
morning
introduced
disgusted
parts
deceptive
dash
mine
funny
opening
photo
confused
fattening
disappointing
depending
promise
ounce
pizzas
pairing
dying
nutrients
course
fills
convinced
cross
microwave
four
kettle
habit
marketed
guess
later
pb
duncan
peppers
lab
eve
prickly
fantastic
florida
eaten
knife
guayaki
purchase
onions
positive
pocket
items
method
french
potato
mom
producing
partially
grocery
marzano
plate
obvious
lovers
protein
gold
expecting
needed
limited
gave
discontinued
looked
metallic
excited
minced
intense
eye
pots
quite
interesting
offered
hunting
match
quantities
insanity
couldnt
happen
palm
hurry
lives
really
european
heaven
produce
lists
pretty
numerous
fabulous
feet
hummus
downside

brand
newman
package
reward
deal
know
line
give
container
deliver
american
may
coffee
coloring
beverage
break
anyway
lb
lid
day
eat
stop
driver
issue
change
color
food
deserve
store
result
grow
sugar
tablespoon
advertising
aftertaste
alternative
aspect
account
turn
allergic
altoids
antioxidants
balance
bigger
active
anymore
bland
beyond
bother
called
bargain
clearly
chocolate
chocolately
boneless
bones
bulk
automatically
breast
channel
al
broke
calories
caution
brewed
black
affect
caramels
bold
choco
breakfast
bill
candies
brush
carries
certified
cayenne
agave
anything
cheaper
appear
chunks
apples
charity
bird
base
colada
cake
aged
areas
adjust
apart
class
blue
boot
bad
bacteria
best
bisque
carbon
cafe
britt
cigars
coffe
celestial
becomes
baseball
adds
bag
addict
closest
cheesecake
ca
balls
afford
audacity
authentic
cinnamon
assortment
books
bite
beginning
bottle
chili
airtight
amount
acceptable
address
audio
castor
candy
aside
caribou
average
brewer
bagged
chicken
chilled
attempt
bodied
additionally
benefits
cases
bowel
began
audible
almost
accident
airport
accidentally
bisquik
caffeinated
blender
cento
colic
care
breading
behind
carbohydrate
changed
caused
classic
children
argentina
coating
caveat
cant
babka
artifical
afterwards
advice
ability
calls
chews
animal
bisquick
clam
chemical
asking
asks
canola
caf
available
breadcereal
batter
cable
absoluteadore

herbs
packaged
kid
sending
kick
firm
improvement
gotten
onto
range
listed
daughter
situation
trusted
friends
0
100
200
300
400
0
100
200
300
400

Number of Clusters 7

In [33]:
```
aw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-2/avgw2vec/7_avgw2vlabels.pkl')
aw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-2/avgw2vec/7_avgw2vwrdcld.pkl')
aw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/task
-2/avgw2vec/7_df_avgw2vlblftname.pkl')
optk = np.unique(aw2vlabels).shape[0]
```

In [34]:
```
for i in range(optk):
    a= list(aw2vdf[aw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)
    if i == 5:
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=8).generate(str1)
        plt.figure(figsize=(10,10))
    else:
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=12).generate(str1)
        plt.figure(figsize=(18,18))
    plt.imshow(wordcld)
```

biscuit
cause
coconut
ear
buck
adult
consumer
celiac
cashew
coat
drop
girl
enhance
human
fit
hour
powder
problem
leftover
point
pound
spot
pancake
natural
purpose
seed
taste
afraid
absolutely
baked
appeal
asked
bottles
aisle
ahoy
additives
across
ad
allergy
ride
banana
anybody
accustomed
tray
absorb
alcohol
seem
alive
second
bitterness
carrot
tip
blend
run
bacon
caviar
alter
age
addicted
shot
baxter
bites
avoid
awful
birthday
back
antioxidant
anti
ask
berry
year
soup
ate
asia
better
boring
chamomile
treat
agreed
burn
canidae
blueberries
cacao
aid
aware
basis
alone
baronet
bummed
admit
babies
already
anywhere
assuming
cakesters
calcium
awhile
remind
breed
bars
berries
beef
afternoon
blame
bliss
bang
celery
buy
apple
becoming
ball
breads
burgers
buyer
salsa
along
twist
acquired
level
wafer
additional
boy
buffet
amazingly
reaction
associated
canister
serving
antibiotics
allowing
barbeque
apparently
arriving
bubble
bees
breaking
blow
cabinet
brother
batches
amazon
canned
appreciate
caramel
aluminum
assorted
asparagus
bbq
body
cane
arrives
appears
blah
brownies
caffeine
broccoli
betty
added
aunt
avoiding
baby
burnt
carry
bob
buster
away
challenge
besides
begin
bring
bank
avid
band
bloody
bringing
buttery
attached
acai
cancelled
attention
came
book
boil
carrying
calling
canadian
brewing
bugs
became
acids
beans
caramel
batches
amazon
aluminum
asparagus
appreciate
aisle
ahoy
arriving
barbeque
apparently
appeal
asked
bottles
bottom
allowing
antibiotics
carefully
serving
barbecue
reaction
associated
canister
burgers
buyer
boy
buffet
breed
celery
ball
breads
acquired
wafer
additional
afternoon
remind
awhile
basis
alone
baronet
bummed
aware
calcium
brainer
certain
box
aid
cacao
burn
blueberries
canidae
ate
ask
berry
birthday
awful
avoid
boil
alter
addicted
shot
baxter
bites
carrying
antioxidant
anti
soup
year
book
came
better
boring
chamomile
treat
agreed
alone
baronet
breed
afternoon
remind
awhile
cakesters
canadian
assuming
anywhere
already
admit
babies
almonds
bars
berries
beef
blame
bliss
bang
celery
buy
apple
becoming
ball
breads
burgers
buyer
salsa
along
twist
acquired
level
wafer
additional
boy
buffet
0
200
400
600
800
1000

form
content
meal
piece
reviewer
vendor
portion
start
alternatives
according
acidic
approximately
belgian
background
anyone
airy
bears
carton
bakery
bottled
consider
colorful
eight
determine
cookbook
glucose
blueberry
beautifully
arrive
grill
familiar
constantly
allow
basic
basically
become
aroma
actual
addicting
calorie
carried
celebrate
cal
drip
flower
bolder
chewy
entire
cheeses
giving
construction
containing
cocker
flat
charged
drinker
chai
comparison
expected
compare
boyfriend
essentials
crap
golf
fight
chain
friend
canin
grease
greatly
dressing
build
complained
china
dented
eater
every
car
cookies
death
buttermilk
everybody
advertised
dr
flavored
extreme
glue
amazing
bee
gatorade
cheapest
certainly
comments
bet
delightful
choose
follows
beware
crappy
grain
allergens
effect
bone
duty
center
checked
daily
goodies
bit
anytime
canada
glycemic
ghiradelli
confection
forever
full
dip
area
dipping
basket
card
free
bonus
goodness
economy
cardboard
gift
experience
fail
brings
checking
burning
edge
grey
beautiful
halal
cooks
gooey
english
beneful
crazy
awesome
fairly
formulas
farmers
creaminess
decafs
flavorless
etc
apricot
groceries
foodie
convenience
carb
earth
folger
boost
fruits
game
bpa
fat
chewing
elsewhere
draw
delicious
cakes
club
coated
foil
extremely
convert
effort
controlled
grapefruit
farms
egg
habanero
fortunes
flour
consider
glucose
cookbook
fillers
costa
equal
couscous
delight
chocolatey
brought
chinese
early
far
disappointed
consuming
department
garden
gums
europe
exact
covered
ginger
0
200
400
600
800
1000

analysis
amazed
affordable
assume
ale
air
block
bluegrass
bull
cocoas
creamers
dioxide
donut
bunch
cocktail
df
dove
drinkable
check
considered
consistency
daughters
distinct
emeril
erin
grove
following
turning
minty
tahini
settled
tictac
stale
sumatra
van
popular
vinegar
roaster
men
paleo
kool
hulls
studio
wellness
gas
seven
gm
sign
mocha
period
theme
trip
puff
thai
handling
scary
wabash
taffy
greatest
filter
look
latest
keurig
gevalia
whirley
slow
guy
gain
time
tully
ups
voted
vegetarian
environment
strong
previously
inhaled
vitacoco
0
200
400
600
800
1000

dissolve
advance
amounts
cents
exotic
cart
cups
onions
dijon
mom
intense
lives
later
obvious
ounce
lists
nutrients
jelly
grass
diced
convinced
pretty
pricy
burned
cheddar
eye
prickly
peppers
fantastic
pretzel
fabulous
cajun
partially
hunting
die
eaten
habit
mountain
guess
funny
instead
one
acts
boxes
arizona
clumped
contains
dash
flaxseeds
morning
marketed
downside
plate
filtered
mine
method
busy
curry
pocket
hurry
protein
fattening
brittle
ended
duncan
ive
marzano
pots
catch
pea
edible
deceptive
organic
pairing
numerous
pizzas
adopted
expecting
memories
missed
raisin
iraq
discontinued
hummus
flowers
french
advanced
palm
bits
member
magnesium
blood
couldnt
pb
evening
eve
european
positive
marinade
florida
depending
happen
promise
opening
packets
parent
cross
grandmother
among
match
attractive
limited
benefit
guayaki
cats
items
claims
cons
offered
knife
minced
fragrance
leaf
kettle
consumed
acid
brownie
big
alpine
feet
quite
really
insanity
produce
course
purchase
call
chance
java
carbs
balanced
looked
gold
excited
bags
olives
grilled
acidity
dying
lab
parts
photo
described
bed
california
mark
quantities
disgusted
peanuts
needed
introduced
brown
disappointing
everywhere
designed
additive
around
country
done
concentrated
heaven
chalky
metallic
carrots
four
interesting
agree
microwave
custard
fills
potato
producing
ara
lovers
matters
grocery
gave
hoped
button
crocker
cooked
confused
congestion
muesli
complaining
earlier
able
low
accessible
0
200
400
600
800
1000

beverage
anyway
american
brand
break
change
coffee
container
deliver
coloring
color
deserve
food
lid
deal
eat
give
issue
newman
lb
reward
sugar
day
grow
driver
store
stop
line
know
turn
may
addict
accident
ability
package
tablespoon
result
assortment
candy
channel
caveat
carries
adds
attempt
affect
chili
college
bottle
anything
coating
anticipated
amount
barley
appear
britt
available
beyond
bigger
base
brush
caution
adore
audio
canola
bitter
caramels
broke
allergic
al
anymore
balance
boys
cinnamon
aftertaste
blue
animalbatter
alternative
chicken
cafe
accidentally
chronic
bacteria
chocolate
apples
bisquik
argentina
caf
account
called
carbohydrate
coffe
afterwards
brew
closest
class
aged
balls
breakfast
benefits
boot
cant
bodied
audible
active
care
castor
began
absolute
best
airport
cases
bill
bite
colada
bargain
cento
bland
agave
black
airtight
almost
carbon
additionally
bagged
bones
bowel
artifical
antioxidants
bold
cereal
beginning
caused
cleaning
chocolately
bulk
aspect
cheaper
audacity
brewed
classic
advice
behind
bisquick
brewer
cable
areas
clean
bread
address
chews
altoids
bad
calls
cheesecake
authentic
badly
bird
afford
apart
books
aside
blender
acceptable
ago
bisque
children
changed
chemical
cake
breast
becomes
automatically
advertising
baseball
colic
charity
bag
average
asks
breading
clearly
candies
babka
ca
clam
chilled
celestial
caribou
cayenne
chunks
calories
cigars
boneless
certified
caffeinated
choco
adjust
asking
bother

improvement
range
daughter
kick
firm
onto
trusted
gotten
kid
packaged
sending
herbs
listed
friends
situation
0
100
200
300
400
0
100
200
300
400

[5.2.3] Applying Agglomerative Clustering on TFIDF W2V, SET 4

In [3]:
```
# Please write all the code with proper documentation
```

In [ ]:
```
        wtw2v =asign10tsk2()
        wtw2v.read_tfidfwtw2vec_dt()
        #print(wtw2v.X_train.shape)
        wtw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
ask-2/wtw2vec/'

        wtw2v.aggclst_tfidfwtw2vec(
```

[5.2.4] Wordclouds of clusters obtained after applying Agglomerative Clustering on TFIDF W2V SET 4

In [3]:
```
# Please write all the code with proper documentation
```

Number of Clusters 2

In [30]:
```
wtw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/task-2/wtw2vec/2_wtw2vlabels.pkl')
wtw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/task-2/wtw2vec/2_wtw2vwrdcld.pkl')
wtw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tas
k-2/wtw2vec/2_df_wtw2vlblftname.pkl')
optk = np.unique(wtw2vlabels).shape[0]
```

In [32]:
```
for i in range(optk):
    a= list(wtw2vdf[wtw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)

    if i == 0 :
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=10).generate(str1)
        plt.figure(figsize=(16,16))
    else:
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=10).generate(str1)
        plt.figure(figsize=(18,18))

    plt.imshow(wordcld)
```

bought
cream
cats
choice
sauce
light
purchased
new
label
hot
even
buy
along
look
spice
meat
getting
care
deal
delivery
difference
quality
eaten
go
oreo
alternative
fine
someone
licorice
several
although
blend
highly
people
subscribe
flavor
coffee
oil
good
soup
really
size
plus
chicken
stars
tasted
enjoyed
always
drink
gf
drinking
fruit
longer
pretty
easily
honey
goes
gluten
meal
butter
often
side
free
service
single
baby
almost
disappointed
stores
chocolate
needed
friends
may
sour
0
100
200
300
400

Number of Clusters 5

In [28]:
```
wtw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/task-2/wtw2vec/5_wtw2vlabels.pkl')
wtw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/task-2/wtw2vec/5_wtw2vwrdcld.pkl')
wtw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tas
k-2/wtw2vec/5_df_wtw2vlblftname.pkl')
optk = np.unique(wtw2vlabels).shape[0]
```

In [29]:
```
for i in range(optk):
    a= list(wtw2vdf[wtw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)

    if i == 3 :
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=12).generate(str1)
        plt.figure(figsize=(18,18))
    else:
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=8).generate(str1)
        plt.figure(figsize=(10,10))
    plt.imshow(wordcld)
```

eat
care
sour
enjoyed
deal
chicken
coffee
difference
disappointed
baby
several
drinking
easily
may
goes
choice
blend
longer
label
fruit
getting
often
0
100
200
300
400
0
100
200
300
400

pretty
highly
gluten
oreo
free
really
light
new
people
service
tasted
value
good
unfortunately
0
100
200
300
400
0
100
200
300
400

single
side
someone
eaten
always
fine
friends
eat
cream
along
even
use
spice
stores
size
honey
cats
flavor
sauce
purchased
quality
meal
meat
almost
soup
gf
subscribe
although
plus
stars
butter
delivery
0
100
200
300
400

food
taste
chip
time
product
do
tea
year
water
coconut
bottle
calorie
brand
cookie
cocoa
find
come
first
better
bag
day
cup
great
flavor
recommend
grocery
pancake
snack
thank
item
anyone
potato
love
sure
bitter
recipe
month
pack
candy
aftertaste
amazon
cost
course
arrived
really
review
thing
make
much
cake
green
old
decaf
actually
week
store
beans
amount
vinegar
problem
ingredient
work
black
know
say
ate
add
seem
awesome
added
ago
cheese
buying
canned
baking
company
baked
hot
absolutely
biscuits
cheaper
energy
artificial
believe
variety
salt
box
cut
another
able
cold
cat
juice
flour
definitely
anything
chocolate
treat
away
available
daughter
everything
far
favorite
feed
flavored
found
crunch
best
fat
fast
health
healthy
gluten
easy
idea
going
husband
last
least
ground
high
house
give
color
delicious
free
fact
feel
excellent
cook
one
extra
good
dark
gum
kids
half
larger
keurig
family
gift
keep
glad
around
brown
cans
bread
amazing
crunchy
bit
cheap
hand
fan
happy
less
heat
couple
etc
dinner
corn
gave
instead
different
hard
dry
cooking
diet
large
fresh
especially
let
every
full
kind
expensive
enough
christmas
even
kettle
bisquick
breakfast
boxes
decided
back
case
crackers
eating
enjoy
big
highly
bad
healthier
came
0
200
400
600
800
1000

Number of Clusters 7

In [25]:
```
wtw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/task-2/wtw2vec/7_wtw2vlabels.pkl')
wtw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluste
r/task-2/wtw2vec/7_wtw2vwrdcld.pkl')
wtw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/tas
k-2/wtw2vec/7_df_wtw2vlblftname.pkl')
optk = np.unique(wtw2vlabels).shape[0]
```

In [27]:
```
for i in range(optk):
    a= list(wtw2vdf[wtw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)

    if i == 3 :
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=12).generate(str1)
        plt.figure(figsize=(18,18))
    else:
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=8).generate(str1)
        plt.figure(figsize=(10,10))

    plt.imshow(wordcld)
```

highly
gluten
good
really
tasted
light
new
service
free
pretty
oreo
people
unfortunately
value
0
100
200
300
400

0
100
200
300
400
0
100
200
300
400
eaten
use

coffee
drinking
blend
sour
easily
choice
several
goes
chicken
0
100
200
300
400
0
100
200
300
400

food
dogs
chip
tea
product
time
water
cookie
tastes
coconut
year
brand
bag
better
day
first
ingredient
potato
great
really
know
make
cup
flavor
bottle
grocery
cocoa
recommend
calorie
find
come
pack
vinegar
review
snack
thank
one
add
black
cooking
awesome
happy
hard
expensive
less
healthier
feed
cake
gluten
cans
beans
juice
diet
cost
different
every
away
salt
sure
seem
another
buying
decaf
old
recipe
pancake
christmas
problem
store
item
bisquick
much
arrived
green
around
flavored
health
canned
came
artificial
easy
back
color
enough
flour
energy
cheese
kettle
enjoy
gum
especially
crunchy
amazon
bit
least
bitter
etc
cheap
amount
anything
corn
healthy
cut
variety
definitely
say
formula
anyone
cheaper
let
treat
half
week
dinner
large
amazing
dark
absolutely
best
house
everyone
brown
fast
free
biscuits
actuallybig
added
ground
home
hand
cat
couple
kind
crackers
gave
idea
aftertaste
fresh
even
fat
glad
everything
husband
larger
got
believe
feel
far
baking
highly
decided
delicious
high
give
excellent
good
heat
keep
full
fan
course
company
ate
chocolate
baked
daughter
keurig
extra
eating
breakfast
cold
dry
instead
box
candy
cook
ago
last
either
work
gift
bad
flavorful
boxes
love
case
kids
able
family
found
fact
hot
available
month
favorite
crunch
bread
going

oil
bought
needed
buy
hot
drink
good
look
alternative
licorice
go
chocolate
0
100
200
300
400
0
100
200
300
400

fruit
deal
label
getting
eat
enjoyed
often
baby
disappointed
care
longer
may
difference
0
100
200
300
400
0
100
200
300
400

[5.3] DBSCAN Clustering

I processed this model with many values for epsilon(EPS). I chose the model that produced the highest number of point that belong to class label 0 and the highest Silhouette score.

For AVGW2V, the optimum eps value is 35

For TFIDFWTW2V, the optimum epsvalue is 0.1

In [ ]:
```
import numpy as np
import pandas as pd
import pickle
import math

from datetime import datetime
from sklearn.cluster import DBSCAN
from sklearn.externals import joblib
from sklearn import metrics

class asign10tsk3 :

        def __init__(self):
                self.X_train =pd.DataFrame()
                self.ipfname = ''
                self.opfname = ''


                self.nclusters = []
                self.dbscan = None
                self.labels = None

                self.w2vecfeat = []
                self.w2vecwrdcld = []

                self.tfidfeat = []
                self.tfidfwrdcld = []

        def read_avgw2vec_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/avgw2vectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'w2vec_feat', 'rb') as fp:
                        self.w2vecfeat = pickle.load(fp)

        def read_tfidfwtw2vec_dt(self):
                self.ipfname = r'E:/appliedaiacourse/assignments/dblite/asi
gn-10-cluster/tfidfwtw2vectorizer/'

                with open(self.ipfname+'ppvectscld_x_train', 'rb') as fp:
                        self.X_train =pickle.load(fp)

                with open(self.ipfname+'tfidf_feat', 'rb') as fp:
                        self.tfidfeat = pickle.load(fp)

        def dbscan_avgw2vec(self):
                all_core_samples =[]
                slh_coef_cntlbl0 = []

                #self.nclusters = [0.1,0.2,0.5,0.7,1,3,4,5,6,7]
                self.nclusters = [10,11,12,13,14,15,16,17,18,19,20,\
                                                                21,22,23,24,25,26,2
7,28,29,30,\
```

```
                                                                31,32,33,34,35,36,3
7,38,39,40] #best any value between 35 and 40(inclsuive of both numbers)

                i = 11

                start_tme=datetime.now()

                for k in self.nclusters:
                        print(k)
                        res_tpl = ()
                        self.dbscan = DBSCAN(eps=k)
                        self.dbscan.fit(self.X_train)
                        self.labels = self.dbscan.labels_

                        core_samp = np.zeros_like(self.labels, dtype=bool)

                        core_samp[self.dbscan.core_sample_indices_] = True
                        all_core_samples.append(core_samp)

                        slh_coef= metrics.silhouette_score(self.X_train, se
lf.labels)

                        avgw2vlblftnme = sorted(zip(self.labels,self.w2vecf
eat))

                        self.w2vecwrdcld = [word for word in avgw2vlblftnme
]
                        df = pd.DataFrame(self.w2vecwrdcld, columns =['lbl'
, 'ftname'])
                        joblib.dump(self.labels,self.opfname+str(i)+'_avgw2
vlabels.pkl')
                        joblib.dump(self.w2vecwrdcld,self.opfname+str(i)+'_
avgw2vwrdcld.pkl')
                        joblib.dump(df,self.opfname+str(i)+'_df_avgw2vlblft
name.pkl')
                        joblib.dump(all_core_samples,self.opfname+str(i)+'a
lll_core_samples.pkl')

                        print('Finished processing cluster:#', k ,' Now: ',
datetime.now(), ' Duration: ', datetime.now()-start_tme)
                        print('Silhouette Coef: ',slh_coef, ' no of point l
bl 0:', len(df[df['lbl']==0].index))
                        print('-'*50)

                        rst_tpl = (slh_coef,len(df[df['lbl']==0].index),sel
f.dbscan.get_params())
                        slh_coef_cntlbl0.append(rst_tpl)
                        joblib.dump(slh_coef_cntlbl0,self.opfname+'slhcof_c
ntlbl0.pkl')
                        print('#'*50)
                        i += 1

        def dbscan_tfidfwtw2vec(self):
                all_core_samples =[]
                slh_coef_cntlbl0 = []

                #self.nclusters = [0.1,0.2,0.5,0.7,1,3,4,5,6,7]
```

```
                self.nclusters = [0.01,0.1,0.2,0.5,0.7,1]
                i = 1

                start_tme = datetime.now()

                for k in self.nclusters:
                        print(k)
                        self.dbscan = DBSCAN(eps=k)
                        self.dbscan.fit(self.X_train)
                        self.labels = self.dbscan.labels_
                        core_samp = np.zeros_like(self.labels, dtype=bool)

                        core_samp[self.dbscan.core_sample_indices_] = True
                        all_core_samples.append(core_samp)

                        slh_coef= metrics.silhouette_score(self.X_train, se
lf.labels)

                        wtw2vlblftnme = sorted(zip(self.labels,self.tfidfea
t))

                        self.tfidfwrdcld = [word for word in wtw2vlblftnme]
                        df = pd.DataFrame(self.tfidfwrdcld, columns =['lbl'
, 'ftname'])
                        joblib.dump(self.labels,self.opfname+str(i)+'_wtw2v
labels.pkl')
                        joblib.dump(self.tfidfwrdcld,self.opfname+str(i)+'_
wtw2vwrdcld.pkl')
                        joblib.dump(df,self.opfname+str(i)+'_df_wtw2vlblftn
ame.pkl')
                        joblib.dump(all_core_samples,self.opfname+str(i)+'a
lll_core_samples.pkl')

                        print('Finished processing cluster:#', k ,' Now: ',
datetime.now(), ' Duration: ', datetime.now()-start_tme)
                        print('Silhouette Coef: ',slh_coef, ' no of point l
bl 0:', len(df[df['lbl']==0].index))
                        print('-'*50)
                        rst_tpl = (slh_coef,len(df[df['lbl']==0].index),sel
f.dbscan.get_params())
                        slh_coef_cntlbl0.append(rst_tpl)
                        print(self.dbscan.get_params())
                        joblib.dump(slh_coef_cntlbl0,self.opfname+'slhcof_c
ntlbl0.pkl')
                        print('#'*50)
                        i += 1

if __name__ == "__main__":


        aw2v =asign10tsk3()
        aw2v.read_avgw2vec_dt()
        print(aw2v.X_train.shape)
        aw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/ta
sk-3/avgw2vec/'

        aw2v.dbscan_avgw2vec()
```

```
        wtw2v =asign10tsk3()
        wtw2v.read_tfidfwtw2vec_dt()
        #print(wtw2v.X_train.shape)
        wtw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/t
ask-3/wtw2vec/'

        wtw2v.dbscan_tfidfwtw2vec()
```

[5.3.1] Applying DBSCAN on AVG W2V, SET 3

In [3]:
```
# Please write all the code with proper documentation
```

In [ ]:
```
        aw2v =asign10tsk3()
        aw2v.read_avgw2vec_dt()
        print(aw2v.X_train.shape)
        aw2v.opfname = 'E:/appliedaiacourse/assignments/asign-10-cluster/ta
sk-3/avgw2vec/'

        aw2v.dbscan_avgw2vec()
```

[5.3.2] Wordclouds of clusters obtained after applying DBSCAN on AVG W2V SET 3

In [2]:
```
# Please write all the code with proper documentation
```

Epsilon 35

In [21]:
```
aw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-3/avgw2vec/36_avgw2vlabels.pkl')
aw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-3/avgw2vec/36_avgw2vwrdcld.pkl')
aw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/task
-3/avgw2vec/36_df_avgw2vlblftname.pkl')
optk = np.unique(aw2vlabels)
print(optk)
print(len(aw2vdf[aw2vdf['lbl']==-1].index))
print(len(aw2vdf[aw2vdf['lbl']==0].index))
```

```
[-1  0  1]
1
3801
```

In [24]:
```
for i in optk:
    a= list(aw2vdf[aw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)

    if i == 0:
        wordcld = WordCloud(width=1000, height=1000, background_color = 'wh
ite',min_font_size=14).generate(str1)
        plt.figure(figsize=(16,16))
    else:
        wordcld = WordCloud(width=500, height=500, background_color = 'whit
e',min_font_size=8).generate(str1)
        plt.figure(figsize=(10,10))

    plt.imshow(wordcld)
```

0
100
200
300
400
0
100
200
300
400

additive
acid
adult
allow
add
age
almond
arrive
cake
bowl
banana
chip
bottle
bread
bee
carb
break
chemical
coloring
flour
gum
day
hit
component
grocer
fat
label
lb
girl
food
guy
drop
include
eat
item
evening
lack
fill
fit
change
choice
chunk
effect
clam
comment
finger
good
jar
fall
egg
lay
flaxseed
hole
know
content
happen
compare
dressing
date
kit
coupon
kind
chew
favor
consumer
kcup
find
grow
animal
drink
calorie
flavour
brand
carrot
comparison
coconut
ingredient
ask
cup
area
infection
caramel
hull
driver
follow
ear
contain
help
cocoa
beverage
brew
gram
deal
flower
amount
ball
drinker
bone
formula
gift
appear
charge
bit
holiday
dog
hate
cracker
horse
eater
cause
bite
bill
con
eye
flavor
give
kibble
enhance
hope
coat
leftover
call
case
deliver
cashew
grain
fact
complaint
cheese
form
dinner
create
juice
bean
alternative
apple
bag
filter
creamer
hold
chicken
favorite
fruit
come
become
deserve
enjoy
cookie
craving
cat
claim
last
brownie
end
hour
fortune
donut
leave
coffee
habit
cook
boy
bar
benefit
biscuit
american
customer
color
diet
fan
concern
human
dessert
hand
celiac
crisp
headache
keep
cereal
chocolate
essential
buck
beat
feel
flavoring
issue
book
anyway
guest
decaf
farm
bring
cost
kernel
dissolve
container
filler
dollar
antioxidant
0
200
400
600
800
1000

[5.3.3] Applying DBSCAN on TFIDF W2V, SET 4

In [3]:
```
# Please write all the code with proper documentation
```

Epsilon 0.1

In [47]:
```
aw2vlabels = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-3/wtw2vec/2_wtw2vlabels.pkl')
aw2vwrdcld = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/
task-3/wtw2vec/2_wtw2vwrdcld.pkl')
aw2vdf = joblib.load('E:/appliedaiacourse/assignments/asign-10-cluster/task
-3/wtw2vec/2_df_wtw2vlblftname.pkl')
optk = np.unique(aw2vlabels)
print(optk)
print(len(aw2vdf[aw2vdf['lbl']==1].index))
print(len(aw2vdf[aw2vdf['lbl']==0].index))
```

```
[-1  0  1  2  3  4  5  6  7  8  9 10]
30
423
```

[5.3.4] Wordclouds of clusters obtained after applying DBSCAN on TFIDF W2V SET 4

In [20]:
```
for i in optk:
    a= list(aw2vdf[aw2vdf.lbl == i]['ftname'])
    str1 = ' '.join(str(e) for e in a)
    if len(str1) >0:
        if i in [0,1]:
            wordcld = WordCloud(width=1000, height=1000, background_color =
'white',min_font_size=14).generate(str1)
            plt.figure(figsize=(16,16))
        else:
            wordcld = WordCloud(width=500, height=500, background_color =
'white',min_font_size=8).generate(str1)
            plt.figure(figsize=(10,10))
    plt.imshow(wordcld)
```

sending
friends
situation
daughter
range
listed
onto
herbs
improvement
firm
kick
gotten
kid
trusted
packaged

green
food
really
potato
vinegar
grocery
half
another
hard
formula
canned
add
believe
keurig
come
tea
one
heat
less
crackers
candy
awesome
boxes
cook
got
even
bad
work
back
coconut
added
artificial
case
cut
salt
baking
best
week
fact
let
amount
cans
decaf
cheap
decided
kind
cost
easy
healthy
much
beans
cold
full
enough
fat
feel
sure
pancake
definitely
find
biscuits
box
enjoy
last
chocolate
pack
store
treat
bit
cat
gluten
bisquick
house
big
energy
fresh
free
keep
bread
everyone
cup
day
better
great
eating
month
away
ate
around
breakfast
fan
healthier
either
crunch
especially
brown
say
recipe
variety
amazon
instead
juice
extra
company
buying
ground
cake
item
flavored
color
gift
home
everything
least
ago
expensive
thank
absolutely
dark
feed
gum
course
good
cookie
bag
crunchy
old
etc
hot
large
arrived
happy
anyone
highly
cheaper
thing
amazing
recommend
bitter
first
flavorful
kettle
fast
idea
anything
gave
delicious
every
corn
family
snack
high
husband
seem
larger
cooking
available
black
flour
aftertaste
able
cheese
found
health
couple
actually
dinner
christmas
going
daughter
make
dry
kids
different
give
baked
review
problem
love
know
water
came
far
favorite
hand
diet
excellent
glad

meat
meal
delivery
size
almost
stars
eat
plus
butter
side
although
gf
cats
fine
even
quality
friends
sauce
honey
single
stores
someone
along
subscribe
soup
always
cream
flavor
purchased
spice
0
200
400
600
800
1000

buy
drink
look
good
licorice
hot
alternative
chocolate
go
bought
oil
needed
0
100
200
300
400
0
100
200
300
400

disappointed
fruit
enjoyed
baby
longer
difference
care
eat
may
label
deal
getting
often
0
100
200
300
400
0
100
200
300
400

choice
several
easily
chicken
coffee
sour
blend
goes
drinking
0
100
200
300
400
0
100
200
300
400

eaten
0
100
200
300
400
0
100
200
300
400

highly
oreo
new
really
pretty
gluten
people
light
value
free
tasted
good
service
unfortunately
0
100
200
300
400
0
100
200
300
400

[6] Conclusions

In [4]:
```
# Please compare all your models using Prettytable library.
# You can have 3 tables, one each for kmeans, agllomerative and dbscan
```

In [50]:
```
head_tab =[['K-Means']]
print(tabulate.tabulate(head_tab,tablefmt='fancy_grid'))
res_tab = [['Model','n_clusters'],
           ['BOW',20],
           ['TFIDF',19],
           ['AVGW2v',20],
           ['TFIDF \nWTW2V',6]
          ]
print(tabulate.tabulate(res_tab,tablefmt='fancy_grid'))
```

```
╒═══════╕
│ K-Means │
╘═══════╛

╒═══════╤═══════╕
│ Model │ n_clusters │
╞═══════╪═══════╡
│ BOW │ 20 │
╞═══════╪═══════╡
│ TFIDF │ 19 │
╞═══════╪═══════╡
│ AVGW2v │ 20 │
╞═══════╪═══════╡
│ TFIDF
│ WTW2V │ 6 │
╘═══════╪═══════╛
```

In [51]:
```
head_tab =[['Agglomorative \nClustering']]
print(tabulate.tabulate(head_tab,tablefmt='fancy_grid'))
res_tab = [['Model','n_clusters','n_clusters','n_clusters'],
           ['AVGW2V',2, 5 ,7],
           ['TFIDF \nWTW2V',2, 5, 7]
          ]
print(tabulate.tabulate(res_tab,tablefmt='fancy_grid'))
```

```
╒═══════╕
│ Agglomorative │
│ Clustering │
╘═══════╛

╒═══════╤═══════╤═══════╤═══════╕
│ Model │ n_clusters │ n_clusters │ n_clusters │
╞═══════╪═══════╪═══════╪═══════╡
│ AVGW2V │ 2 │ 5 │ 7 │
╞═══════╪═══════╪═══════╪═══════╡
│ TFIDF
│ WTW2V │ 2 │ 5 │ 7 │
╘═══════╪═══════╪═══════╪═══════╛
```

In [48]:
```
head_tab =[['DBSCAN']]
print(tabulate.tabulate(head_tab,tablefmt='fancy_grid'))
res_tab = [['Model','Epsilon','Count of\n Words Label 1','Count of\n Words
 Label 0'],
          ['AVGW2V',35, 1 ,3801],
          ['TFIDFWTW2V',0.1, 30, 423]
         ]
print(tabulate.tabulate(res_tab,tablefmt='fancy_grid'))
```

```
╒═══════╕
│ DBSCAN │
╘═══════╛

╒═══════════╤═══════╤═══════════╤═══════════╕
│ Model     │ Epsilon │ Count of  │ Count of  │
│           │         │ Words Label 1 │ Words Label 0 │
├───────────┼─────────┼───────────┼───────────┤
│ AVGW2V    │ 35      │ 1         │ 3801      │
├───────────┼─────────┼───────────┼───────────┤
│ TFIDFWTW2V │ 0.1     │ 30        │ 423       │
╘═══════════╧═══════╧═══════════╧═══════════╛
```