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 [62]:

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


import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os
```

In [63]:

```
# using SQLite Table to read data.
con = sqlite3.connect('C:\\Users\\sujpanda\\Desktop\\applied\\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 a
# you can change the number to any other number based on your computing power

# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score !=
# for tsne assignment you can take 5k data points

filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3

# Give reviews with Score>3 a positive rating(1), and reviews with a score<3
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[63]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominat
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	

In [64]:

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

In [65]:
```
print(display.shape)
display.head()
```

(80668, 7)

Out[65]:

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)
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 [66]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[66]:

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

In [67]:
```
display['COUNT(*)'].sum()
```

Out[67]: 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 [68]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[68]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomir
0	78445	B000HDL1RQ	AR5J8UI46CURR	Geetha Krishnan	2	
1	138317	B000HDOPYC	AR5J8UI46CURR	Geetha Krishnan	2	
2	138277	B000HDOPYM	AR5J8UI46CURR	Geetha Krishnan	2	
3	73791	B000HDOPZG	AR5J8UI46CURR	Geetha Krishnan	2	
4	155049	B000PAQ75C	AR5J8UI46CURR	Geetha Krishnan	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 [69]:
```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, ir
```

In [70]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text
final.shape
```

Out[70]: (4986, 10)

In [71]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[71]: 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 [72]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[72]:

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

In [73]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [74]:
```
#Before starting the next phase of preprocessing lets see the number of entri
print(final.shape)

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

```
(4986, 10)
```

Out[74]:
```
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 [75]:

```python
# 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 />htt
p://www.amazon.com/VICTOR-FLY-MAGNET-BAIT-REFILL/dp/B00004RBDY<br /><br />The V
ictor M380 and M502 traps are unreal, of course -- total fly genocide. Pretty s
tinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious these chi
ps are.  The best thing was that there were a lot of "brown" chips in the bsg
(my favorite), so I bought some more through amazon and shared with family and
friends.  I am a little disappointed that there are not, so far, very many brow
n chips in these bags, but the flavor is still very good.  I like them better t
han 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 re
commend that you try a bag before buying bulk.  They 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 were o
rdering; the other wants crispy cookies.  Hey, I'm sorry; but these reviews do
nobody any good beyond reminding us to look  before ordering.<br /><br />These
are chocolate-oatmeal cookies.  If you don't like that combination, 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 c
onsistency.  Now let's also remember that tastes differ; so, I've given my opin
ion.<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 don't see where these taste like ra
w cookie dough.  Both are soft, however, so is this the confusion?  And, yes, t
hey stick together.  Soft cookies tend to do that.  They aren't individually wr
apped, 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 Snaps.  If you want a cookie that's soft, chewy and tastes like
a combination 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 [76]:

```
# 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 genocid
e. Pretty stinky, but only right nearby.
```

In [77]:
```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-re
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? />The Vi
ctor M380 and M502 traps are unreal, of course -- total fly genocide. Pretty st
inky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious these chi
ps are.  The best thing was that there were a lot of "brown" chips in the bsg
(my favorite), so I bought some more through amazon and shared with family and
friends.  I am a little disappointed that there are not, so far, very many brow
n chips in these bags, but the flavor is still very good.  I like them better t
han 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 re
commend that you try a bag before buying bulk.  They 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 were o
rdering; the other wants crispy cookies.  Hey, I'm sorry; but these reviews do
nobody any good beyond reminding us to look  before ordering.These are chocolat
e-oatmeal cookies.  If you don't like that combination, don't order this type o
f cookie.  I find the combo quite nice, really.  The oatmeal sort of "calms" th
e rich chocolate flavor and gives the cookie sort of a coconut-type consistenc
y.  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 coo
kie dough; however, I don't 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 aren't individually wrapped, which would add
to the cost.  Oh yeah, chocolate chip cookies tend to be somewhat sweet.So, if
you want something hard and crisp, I suggest Nabiso's Ginger Snaps.  If you wan
t a cookie that's soft, chewy and tastes like a combination of chocolate and oa
tmeal, give these 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 gre
at coffee.  dcaf is very good as well
```

In [78]:

```
# 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 [79]:

```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Wow.  So far, two two-star reviews.  One obviously had no idea what they were o
rdering; the other wants crispy cookies.  Hey, I am sorry; but these reviews do
nobody 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 ord
er 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-typ
e consistency.  Now let is also remember that tastes differ; so, I have given m
y opinion.<br /><br />Then, these are soft, chewy cookies -- as advertised.  Th
ey are not "crispy" cookies, or the blurb would say "crispy," rather than "chew
y."  I happen to like raw cookie dough; 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.  Soft cookies tend to do that.  They are not individu
ally wrapped, which would add to the cost.  Oh yeah, chocolate chip cookies ten
d to be somewhat sweet.<br /><br />So, if you want something hard and crisp, I
suggest Nabiso is Ginger Snaps.  If you want a cookie that is soft, chewy and t
astes like a combination of chocolate and oatmeal, give these a try.  I am here
to place my second order.
==================================================
```

In [80]:

```
#remove words with numbers python: https://stackoverflow.com/a/18082370/40840
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 genocide. Prett
y stinky, but only right nearby.
```

In [81]:
```
#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 orderi
ng the other wants crispy cookies Hey I am sorry but these reviews do nobody an
y good beyond reminding us to look before ordering br br These are chocolate oa
tmeal cookies If you do not like that combination do not order this type of coo
kie I find the combo quite nice really The oatmeal sort of calms the rich choco
late 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 ar
e soft chewy cookies as advertised They are not crispy cookies or the blurb wou
ld say crispy rather than chewy I happen to like raw cookie dough however I do
not see where these taste like raw cookie dough Both are soft however so is thi
s the confusion And yes they stick together Soft cookies tend to do that They a
re not 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 is Ginger Snaps If you want a cookie that is soft chewy and ta
stes like a combination of chocolate and oatmeal give these a try I am here to
place my second order
```

In [82]:
```
# 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 1s

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours',
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'i
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'beca
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'th
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shou
            'won', "won't", 'wouldn', "wouldn't"])
```

In [83]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(final['Text'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 4986/4986 [00:02<00:00, 2091.11it/s]
```

In [84]:
```
preprocessed_reviews[1500]
```

Out[84]:
```
'wow far two two star reviews one obviously no idea ordering wants crispy cooki
es hey sorry reviews nobody good beyond reminding us look ordering chocolate oa
tmeal cookies not like combination not order type cookie find combo quite nice
really oatmeal sort calms rich chocolate flavor gives cookie sort coconut type
consistency let also remember tastes differ given opinion soft chewy cookies ad
vertised not crispy cookies blurb would say crispy rather chewy happen like raw
cookie dough however not see taste like raw cookie dough soft however confusion
yes stick together soft cookies tend not individually wrapped would add cost oh
yeah chocolate chip cookies tend somewhat sweet want something hard crisp sugge
st nabiso ginger snaps want cookie soft chewy tastes like combination chocolate
oatmeal give try place second order'
```

[3.2] Preprocessing Review Summary

In [85]:
```
from tqdm import tqdm
preprocessed_summary = []
# tqdm is for printing the status bar
for sentance in tqdm(final['Summary'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not
    preprocessed_summary.append(sentance.strip())
```

```
100%|██████████| 4986/4986 [00:01<00:00, 2812.55it/s]
```

[4] Featurization

[4.1] BAG OF WORDS

In [86]:

```
#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', 'abbott', 'a
bby', '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 [87]:

```
#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/st

# 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=5000)
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 ", fina
```

```
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 [88]:
```
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_feat
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 ", fina
```

```
some sample features(unique words in the corpus) ['ability', 'able', 'able fin
d', 'able get', 'absolute', 'absolutely', 'absolutely delicious', 'absolutely l
ove', '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 [89]:
```
# 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 [90]:

```
# 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/#.W17SRFAzZ
# 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-negat
        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_w2v
```

```
[('excellent', 0.9952961802482605), ('especially', 0.9938456416130066), ('regul
ar', 0.9937968850135803), ('amazing', 0.9937533140182495), ('calorie', 0.993622
4818229675), ('overall', 0.993571937084198), ('healthy', 0.9934845566749573),
('alternative', 0.9933762550354004), ('looking', 0.9933747053146362), ('snack',
0.9933668971061707)]
==================================================
[('varieties', 0.9994357228279114), ('eaten', 0.9994240403175354), ('stash', 0.
9994170069694519), ('beef', 0.9993213415145874), ('become', 0.999253749847412
1), ('particularly', 0.9992130994796753), ('somewhat', 0.9992072582244873), ('d
e', 0.9991826415061951), ('remember', 0.9991679191589355), ('bar', 0.9991577267
64679)]
```

In [91]:
```
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', 'stinky',
'right', 'nearby', 'used', 'ca', 'not', 'beat', 'great', 'received', 'shipmen
t', 'could', 'hardly', 'wait', 'try', 'love', 'call', 'instead', 'removed', 'ea
sily', 'daughter', 'designed', 'printed', 'use', 'car', 'windows', 'beautifull
y', 'shop', 'program', 'going', 'lot', 'fun', 'everywhere', 'like', 'tv', 'comp
uter', '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 [92]:
```
# average Word2Vec
# compute average word2vec for each review.
sent_vectors = []; # the avg-w2v for each sentence/review is stored in this l
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 migh
    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%|██████████| 4986/4986 [00:05<00:00, 850.00it/s]

4986
50
```

[4.4.1.2] TFIDF weighted W2v

In [93]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
tf_idf_matrix = model.fit_transform(preprocessed_reviews)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))
```

In [94]:
```
# 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_val

tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored i
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/review
    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 [00:38<00:00, 130.75it/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)](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.

some utility functions

In [95]:

```
def printWordCloud(df,input=26):
    import matplotlib.pyplot as plt
    from wordcloud import WordCloud, STOPWORDS
    i = 0
    for i in range(input):
        wordslist = df[df['labels'] == i]['desc']
        print("word list for label = ",i)
        print(wordslist)
        if not wordslist.values[0]:
            print("Word cloud is not supported for label : ",i)
            continue
        if(len(wordslist) > 0):
            stringlist = " ".join(wordslist)
            wordcloud = WordCloud(relative_scaling = 1.0,
                        stopwords = set(STOPWORDS)
                        ).generate(stringlist)
            plt.imshow(wordcloud)
            plt.axis("off")
            plt.title("cluster number " + str(i))
            plt.show()
```

In [96]:

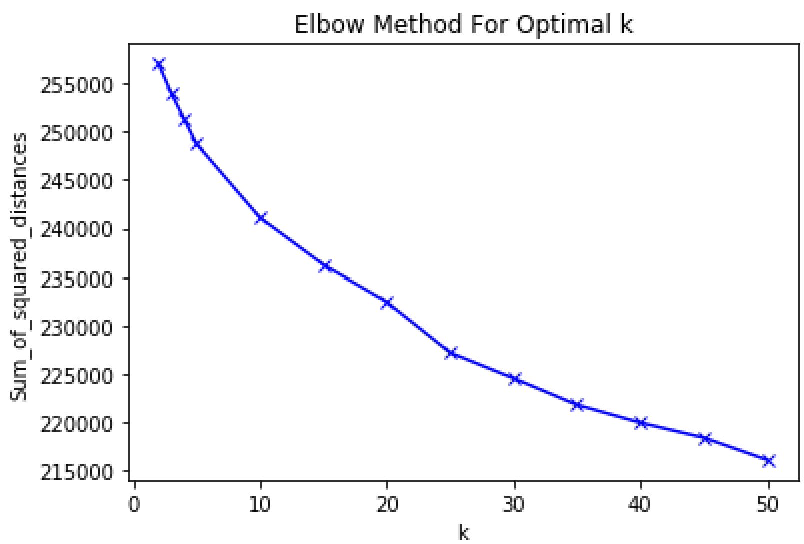
```
def kmeanWithK(vects):
    from sklearn.cluster import KMeans
    Sum_of_squared_distances = []
    K = [2,3,4,5,10,15,20,25,30,35,40,45,50]
    for k in K:
        km = KMeans(n_clusters=k)
        km = km.fit(vects)
        Sum_of_squared_distances.append(km.inertia_)


    plt.plot(K, Sum_of_squared_distances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Sum_of_squared_distances')
    plt.title('Elbow Method For Optimal k')
    plt.show()
```

In [97]:
```
def aggloWihtK(vects):
    from sklearn.cluster import AgglomerativeClustering
    from sklearn.metrics import silhouette_samples, silhouette_score
    avg_score = []
    K = [2,3,4,5,10,15,20,25,30,35,40,45,50]
    for k in K:
        agg = AgglomerativeClustering(n_clusters=k)
        pred = agg.fit_predict(vects)
        silhouette_avg = silhouette_score(vects, pred)
        print("For n_clusters =", k,"The average silhouette_score is :", silh
        avg_score.append(silhouette_avg)
    print("The max silhouette_avg is ",max(avg_score))
```

[5.1] K-Means Clustering

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

In [98]:
```
kmeanWithK(final_counts)
```



In [100]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=25, random_state=0).fit(final_counts)
```

In [101]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
```

In [102]:
```
labels = kmeans.labels_
```

In [103]:
```
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
```

In [104]:
```
df.head()
```

Out[104]:

	desc	labels
0	product available victor traps unreal course t...	3
1	used victor fly bait seasons ca not beat great...	3
2	received shipment could hardly wait try produc...	3
3	really good idea final product outstanding use...	3
4	glad cocker standard poodle puppy loves stuff ...	3

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

In [105]:
```
printWordCloud(df,25)
```

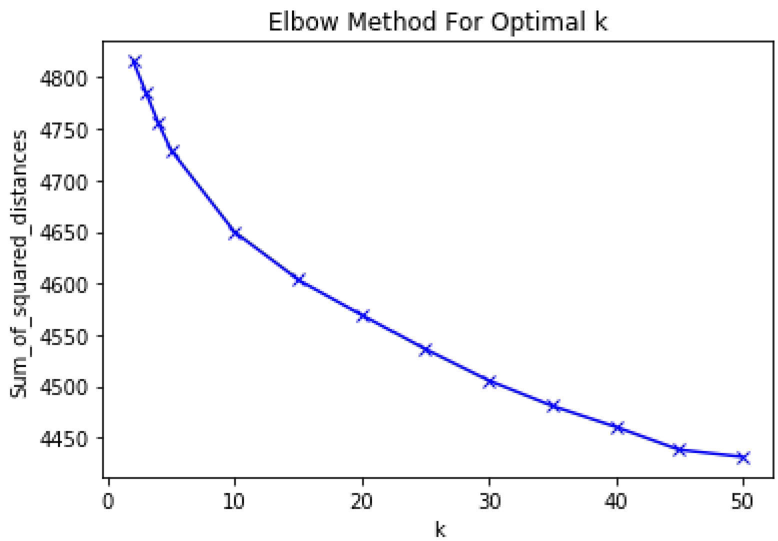
```
4972    love dont use much strong tastes great makes e...
4973    best olive oil not cooking olive oil work flav...
4974    nearby fresh easy neighborhood market stocks n...
4975                                  get walmart gas station
4976    coffee really rich perfect morning ordered off...
4977    fresh great way get little chocolate life with...
4979    wonderful dinner fresh fluke fried tempura bat...
4983    bold blend great taste flavor comes bursting u...
4984    coffee available tassimo kona richest flavor f...
Name: desc, Length: 2643, dtype: object
```

cluster number 3

Observation: Cluster 0 : Talks about food and ingerdients, Cluster 1 : Clubs the taste of the food into one category,Cluster 2: Catgorizes coffee related reviews,Cluster 3 : Caterorizes best products reviews

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

In [106]:
```
kmeanWithK(final_tf_idf)
```



In [107]:
```
kmeans = KMeans(n_clusters=25, random_state=0).fit(final_tf_idf)
```

In [108]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
labels = kmeans.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

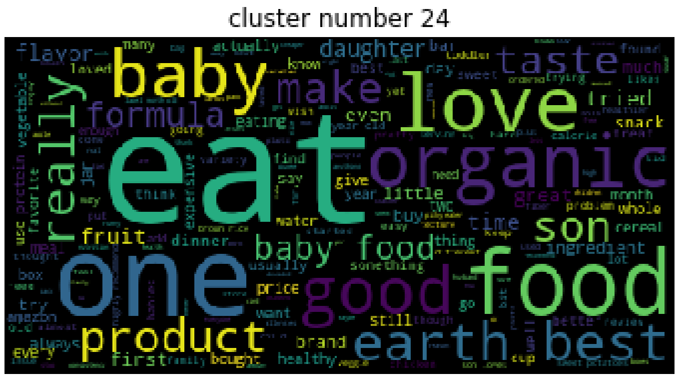
Out[108]:

	desc	labels
0	product available victor traps unreal course t...	3
1	used victor fly bait seasons ca not beat great...	3
2	received shipment could hardly wait try produc...	3
3	really good idea final product outstanding use...	3
4	glad cocker standard poodle puppy loves stuff ...	17

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

In [109]:
```
printWordCloud(df,25)
```

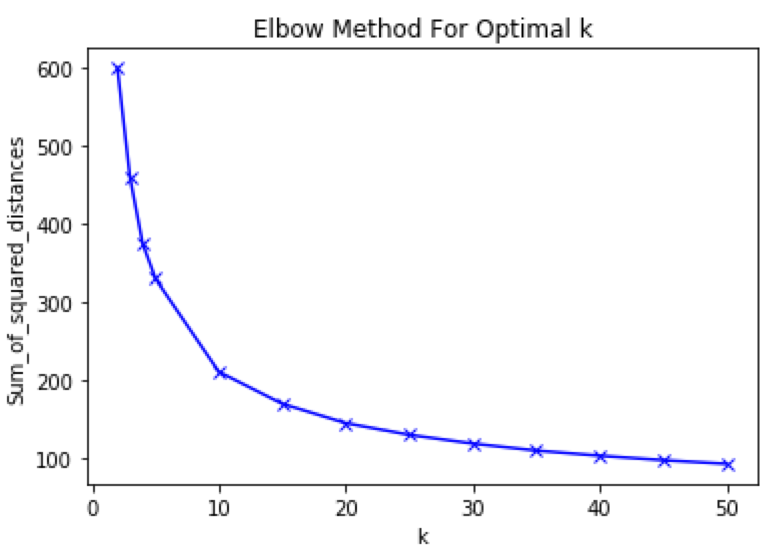
```
4726    old son not picky eater first refused one flav...
4727    baby loves dinners flavorful not bland like ba...
4728    love earth best son eats almost earth best sin...
4729    organic foods course best health starting baby...
4752    peppermint stick delicious fun eat dad got one...
Name: desc, Length: 177, dtype: object
```



Observation: Label 0 : Coffee related reviews,label1 = Chips ad its taste,cluster 21: Taste of food related.Cluster 24 : Baby food related reviews.

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

In [110]:
```
kmeanWithK(sent_vectors)
```



In [111]:
```
kmeans = KMeans(n_clusters=15, random_state=0).fit(sent_vectors)
```

In [112]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
labels = kmeans.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

Out[112]:

	desc	labels
0	product available victor traps unreal course t...	6
1	used victor fly bait seasons ca not beat great...	7
2	received shipment could hardly wait try produc...	2
3	really good idea final product outstanding use...	2
4	glad cocker standard poodle puppy loves stuff ...	2

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

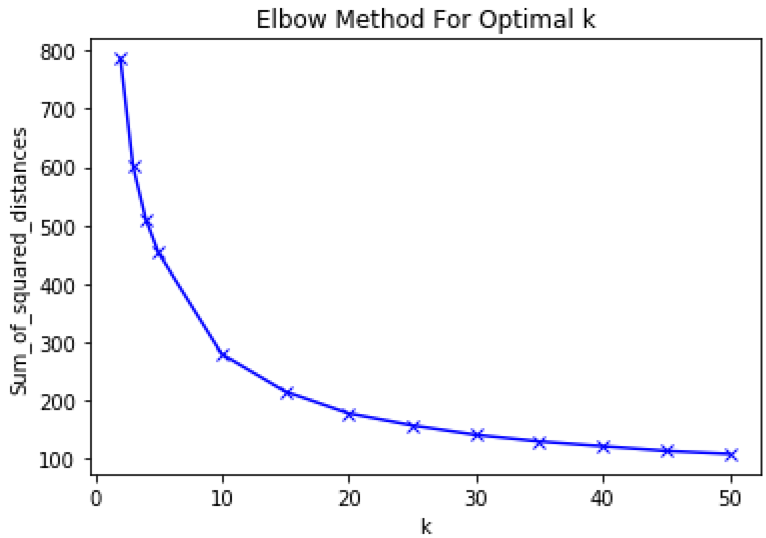
In [113]:
```
printWordCloud(df,15)
```

```
word list for label =  6
0       product available victor traps unreal course t...
6       nine cats crazy kibbles last thing want cat fo...
15      thank goodness mexgrocer love pico pica sauce ...
16      different sauce nothing like anything find gro...
```

Observation: Cluster 1 : Categorized related to taste,cluster 3 categorized related to bakery items

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

In [114]:
```
kmeanWithK(tfidf_sent_vectors)

```



In [115]:
```
kmeans = KMeans(n_clusters=15, random_state=0).fit(tfidf_sent_vectors)
```

In [116]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
labels = kmeans.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

Out[116]:

	desc	labels
0	product available victor traps unreal course t...	10
1	used victor fly bait seasons ca not beat great...	14
2	received shipment could hardly wait try produc...	5
3	really good idea final product outstanding use...	5
4	glad cocker standard poodle puppy loves stuff ...	10

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

In [117]:
```
printWordCloud(df,15)
```

```
154     started using nustevia several years ago back ...
165     nunaturals stevia sweetener use beverages cook...
186     dog loved think probably helped gas dare say b...
188     cardigan welsh corgi loves things stopped carr...
189     went searching charcoal dog biscuits reading h...
203     product arrived timely manner good condition h...
215     expecting cookies going normal size wrong plan...
219     cookies kind stail fast shipping good packing ...
222     great product fast shipment food product taste...
226     pleased product almost not buy looked green co...
255     enjoyed product also provided fast shipping ne...
256     mix great anytime love would recommend anyone ...
270     canidae felidae also changed formula cats not ...
274     used types baking including cakes biscuits coo...
276     decided try formula baby since started gassy h...
                                ...
4671    excellent tasty though thought not enough garl...
4684    bites enough snack delicious price reasonable ...
4690    purchased product greatly disappointed product...
4698    one year old loves product eats one every day ...
```

Observation: Cluster 1 related to beverages,cluster 2: Related to bakery items

[5.2] Agglomerative Clustering

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

In [118]:
```
from sklearn.cluster import AgglomerativeClustering
aggloWihtK(sent_vectors)
```

```
For n_clusters = 2 The average silhouette_score is : 0.24122693308014306
For n_clusters = 3 The average silhouette_score is : 0.2564291818223079
For n_clusters = 4 The average silhouette_score is : 0.26591183937602414
For n_clusters = 5 The average silhouette_score is : 0.19072332977732834
For n_clusters = 10 The average silhouette_score is : 0.15635637747987344
For n_clusters = 15 The average silhouette_score is : 0.11618735844944306
For n_clusters = 20 The average silhouette_score is : 0.12241657607798785
For n_clusters = 25 The average silhouette_score is : 0.11660228949225729
For n_clusters = 30 The average silhouette_score is : 0.11562904660639953
For n_clusters = 35 The average silhouette_score is : 0.11813084568875697
For n_clusters = 40 The average silhouette_score is : 0.11692284970932561
For n_clusters = 45 The average silhouette_score is : 0.1086335101492113
For n_clusters = 50 The average silhouette_score is : 0.10842209682685071
The max silhouette_avg is  0.26591183937602414
```

In [119]:
```
agg = AgglomerativeClustering(n_clusters=3).fit(sent_vectors)
```

In [120]:
```python
newproce_reviews = np.asarray(preprocessed_reviews)
labels = agg.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

Out[120]:

	desc	labels
0	product available victor traps unreal course t...	0
1	used victor fly bait seasons ca not beat great...	2
2	received shipment could hardly wait try produc...	0
3	really good idea final product outstanding use...	0
4	glad cocker standard poodle puppy loves stuff ...	0

In [121]:
```python
df.labels.unique()
```

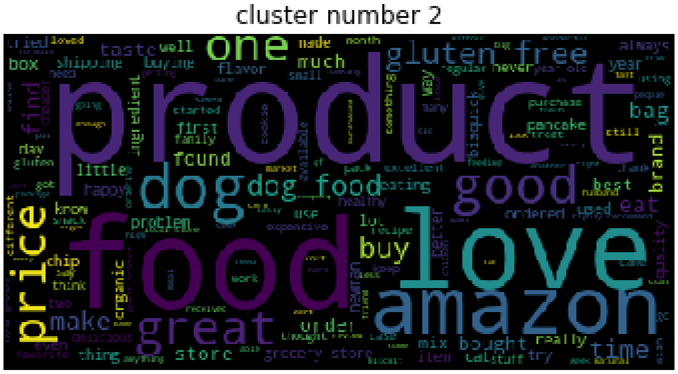
Out[121]: array([0, 2, 1], dtype=int64)

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

In [122]:
```python
printWordCloud(df,3)
```

```
4882    shipping great loved product boxes little toug...
4884    love juice scheduled shipment convenient econo...
4966    never sunchy malta drank lot goya available lo...
4967    item got house time surprised well package pac...
4968    since gluten free tried types gf breads expens...
Name: desc, Length: 930, dtype: object
```

cluster number 2




```
Observation: Cluster 0 : Taste,Cluster 1: Grocery related,cluster 2 :
Amimal and personal food choice
```

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

In [123]:
```
agglоWihtK(tfidf_sent_vectors)
```

```
For n_clusters = 2 The average silhouette_score is : 0.27948948191504586
For n_clusters = 3 The average silhouette_score is : 0.2750057768563206
For n_clusters = 4 The average silhouette_score is : 0.2818525993789562
For n_clusters = 5 The average silhouette_score is : 0.22365663758884447
For n_clusters = 10 The average silhouette_score is : 0.16642464750728078
For n_clusters = 15 The average silhouette_score is : 0.1558903168346997
For n_clusters = 20 The average silhouette_score is : 0.14872175951287364
For n_clusters = 25 The average silhouette_score is : 0.1523530876563093
For n_clusters = 30 The average silhouette_score is : 0.14403880435020086
For n_clusters = 35 The average silhouette_score is : 0.13179144324220163
For n_clusters = 40 The average silhouette_score is : 0.12500105292529806
For n_clusters = 45 The average silhouette_score is : 0.12735163699228913
For n_clusters = 50 The average silhouette_score is : 0.13000658899725717
The max silhouette_avg is  0.2818525993789562
```

In [124]:
```
agg = AgglomerativeClustering(n_clusters=4).fit(tfidf_sent_vectors)
```

In [125]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
labels = agg.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

Out[125]:

	desc	labels
0	product available victor traps unreal course t...	1
1	used victor fly bait seasons ca not beat great...	1
2	received shipment could hardly wait try produc...	1
3	really good idea final product outstanding use...	1
4	glad cocker standard poodle puppy loves stuff ...	1

In [126]:
```
print(df.labels.unique())
```

```
[1 2 0 3]
```

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

In [127]:
```
printWordCloud(df,4)
```

```
word list for label =  3
24
121
456
601
718
792
820     like plockysthey like us plockys plockys mean ...
1042
1136
1183
1272
1781
1816
2023
2107
2353
Name: desc, dtype: object
Word cloud is not supported for label :  3
```

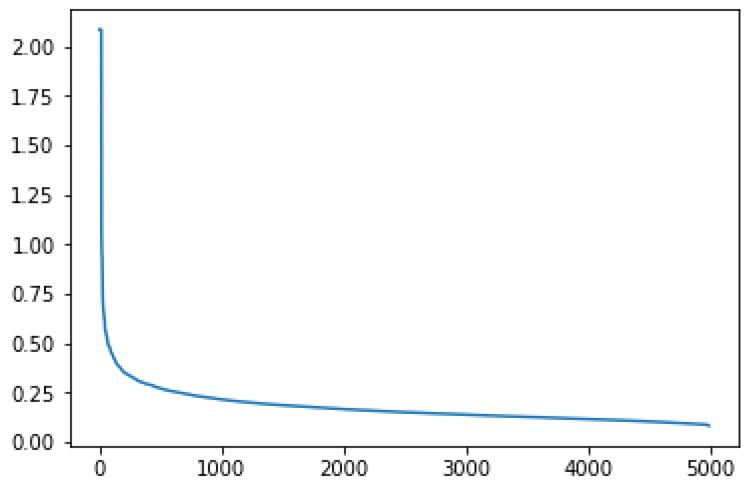
Observation: Cluster 0 : Taste,Cluster 1: Grocery related,cluster 2 : Amimal and personal food choice

[5.3] DBSCAN Clustering

[5.3.1] Applying DBSCAN on AVG W2V, SET 3

In [128]:
```
from sklearn.neighbors import NearestNeighbors
ns = 100
nbrs = NearestNeighbors(n_neighbors=ns).fit(sent_vectors)
distances, indices = nbrs.kneighbors(sent_vectors)
distanceDec = sorted(distances[:,ns-1], reverse=True)

plt.plot(list(range(1,4986+1)), distanceDec)
```

Out[128]: [<matplotlib.lines.Line2D at 0xc400278>]



In [129]:
```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=0.30, min_samples=100).fit(sent_vectors)
```

In [130]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
labels = clustering.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

Out[130]:

	desc	labels
0	product available victor traps unreal course t...	0
1	used victor fly bait seasons ca not beat great...	0
2	received shipment could hardly wait try produc...	0
3	really good idea final product outstanding use...	0
4	glad cocker standard poodle puppy loves stuff ...	0

In [131]:
```
print(df.labels.unique())
```

```
[ 0 -1]
```

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

In [132]:

```python
wordslist = df[df['labels'] == -1]['desc']
print(wordslist)
if(len(wordslist) > 0):
    stringlist = " ".join(wordslist)
    wordcloud = WordCloud(relative_scaling = 1.0,
                          stopwords = set(STOPWORDS)
                          ).generate(stringlist)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.title("cluster number " + str(-1))
    plt.show()
```

```
24
121
378     sooooo deliscious bad ate em fast gained pds f...
381         tasty gluten free option kids loved not spicy
456
601
718
792
903        salt free product purchased chips quite greasy
1042
1057    price product certainly raises attention compa...
1127    licorice good taste daughter gluten free diet ...
1136
1183
1226    would recommend product received timely manner...
1272
1485    son gluten free not fun tasty free great puthi...
1528    good product decent shipping however price lat...
1648        using product year goldie dog loves years old
1781
1816
1918    great dog food dog severs allergies brand one ...
1943    dog loves loves dog food say love twice dog fo...
2023
2107
2307    hard find locally great price especially free ...
2353
2506    using couple years highly recommend need cup c...
2509                                               coffee
2526    food gets rating compared failing scores many ...
                              ...
3821    never made gluten free items got great reviews...
3827    product similar taste wheat pancakes without g...
3837    great product hats betty crocker becoming main...
3846    family tried every gluten free pancake mix mar...
3851    love good makes perfect pancakes love also dai...
3864    recently go gluten free disappointed gluten fr...
3866    discovered sensitivity gluten months ago disma...
3883    absolute best baking product kind tried almost...
3885    love product gluten free products come long wa...
3888    makes good tasting gluten free food like glute...
3893    not find better gluten free pancake mix matter...
3897    great product us gluten free people use everyt...
3898    grateful amazing mix used make best gluten fre...
3899    tried many gluten free products market far eas...
```

```
3913    versital product love husband must eat gluten ...
3915    served gluten free bisquick pancake waffle mix...
3924    product great pancakes baked goods blend bisqu...
3930    use gluten free bisquick make coated baked chi...
3945    received item time packaging great gluten free...
3947    no idea gluten free bisquick mix made pancakes...
3986    could not find gum grocery store ordered amazo...
4024    wonderful dark chocolate flavor much deeper fl...
4337    great bargain worked cheaper purchased wholesa...
4360    really tasty hot chocolate like much dark choc...
4410    delicious hot chocolate best oz cup stretched ...
4459    not sweet best k cup hot chocolate tried thus ...
4497    hot chocolate much better k cup hot chocolates...
4541    far best k cup hot cocoa tried nice rich flavo...
4577    liked milk chocolate version tried good intens...
4586    grove square hot cocoa right amout milk chocol...
Name: desc, Length: 78, dtype: object
```

In [133]:
```
wordslist = df[df['labels'] == 0]['desc']
print(wordslist)
if(len(wordslist) > 0):
    stringlist = " ".join(wordslist)
    wordcloud = WordCloud(relative_scaling = 1.0,
                          stopwords = set(STOPWORDS)
                          ).generate(stringlist)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.title("cluster number " + str(0))
    plt.show()
```

```
0       product available victor traps unreal course t...
1       used victor fly bait seasons ca not beat great...
2       received shipment could hardly wait try produc...
3       really good idea final product outstanding use...
4       glad cocker standard poodle puppy loves stuff ...
5       using food months find excellent fact two dogs...
6       nine cats crazy kibbles last thing want cat fo...
7       shipped day ordered arrived within days live o...
8       mix probably not something would want use ever...
9       description product disceptive product represe...
10      bought brand online indian grocery store usual...
11      use keep finicky toddler protein levels great ...
12      get busy home like sausage lot quick meal opti...
13      company american classic business years best h...
14      love pico pica adds flavor not hot eat least m...
15      thank goodness mexgrocer love pico pica sauce ...
16      different sauce nothing like anything find gro...
17      found product search edible gold leaf decided ...
18      purchased item cake called gold dust never tho...
19      used product multiple times fact purchased fou...
20      used super gold luster dust create exquisite c...
21      product allows make really big splashes provid...
22      cute affordable set son golf theme party great...
23      used one green ball etc golfer cake made one l...
25      golf set arrived quickly pictured birthday fiv...
26      natural ingredients no preservatives say fanta...
27      adzuki azuki beans ment used asian sweets make...
28      good beans not find grocery stores live ordere...
29      not good yummy smell like cloves cooking taste...
30      good stuff like lentils not need soak small fe...
                              ...
4956    carabou mahogony worst tasting cup coffee ever...
4957    ordered mahogany caribou coffee k cups adore r...
4958    wife avid keurig coffee fans three brewers two...
4959    serious cup joe yummyness turned least friends...
4960    maybe greatest coffee ever made many time regu...
4961    surprised find taiwan shaped pineapple cakes c...
4962    really like pineapple shortcakes sold unlike m...
4963                                        started using
4964    using lourdes chimichurri years stuff awesome ...
4965    absolutely love product use chicken beef veggi...
4966    never sunchy malta drank lot goya available lo...
4967    item got house time surprised well package pac...
4968    since gluten free tried types gf breads expens...
4969    not good houston samba grill tasted great text...
```

```
4970    made recently holiday party never seen anythin...
4971    spent first five years life brazil american pa...
4972    love dont use much strong tastes great makes e...
4973    best olive oil not cooking olive oil work flav...
4974    nearby fresh easy neighborhood market stocks n...
4975                                  get walmart gas station
4976    coffee really rich perfect morning ordered off...
4977    fresh great way get little chocolate life with...
4978    taste something like flax bread cornbread exce...
4979    wonderful dinner fresh fluke fried tempura bat...
4980    tried available discs kona blend one best regu...
4981    one best choices opinion also adore amazon nee...
4982    tried many tassimo flavors far favorite normal...
4983    bold blend great taste flavor comes bursting u...
4984    coffee available tassimo kona richest flavor f...
4985    coffee supposedly premium tastes watery thin n...
Name: desc, Length: 4908, dtype: object
```

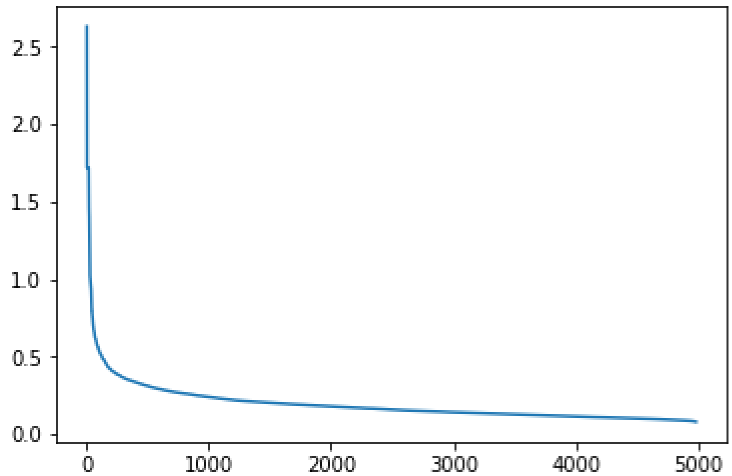
cluster number 0

Observation: Cluster -1 : Bakery,cluster 1 : Categorized related to taster

[5.3.3] Applying DBSCAN on TFIDF W2V, SET 4

In [134]:
```
from sklearn.neighbors import NearestNeighbors
ns = 100
nbrs = NearestNeighbors(n_neighbors=ns).fit(tfidf_sent_vectors)
distances, indices = nbrs.kneighbors(tfidf_sent_vectors)
distanceDec = sorted(distances[:,ns-1], reverse=True)

plt.plot(list(range(1,4986+1)), distanceDec)
```

Out[134]: `[<matplotlib.lines.Line2D at 0x12f0d4a8>]`



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

In [135]:
```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=0.3, min_samples=100).fit(tfidf_sent_vectors)
```

In [136]:
```
newproce_reviews = np.asarray(preprocessed_reviews)
labels = clustering.labels_
df = pd.DataFrame({'desc':newproce_reviews, 'labels':labels})
df.head()
```

Out[136]:

	desc	labels
0	product available victor traps unreal course t...	0
1	used victor fly bait seasons ca not beat great...	0
2	received shipment could hardly wait try produc...	0
3	really good idea final product outstanding use...	0
4	glad cocker standard poodle puppy loves stuff ...	0

In [137]:
```
print(df.labels.unique())
```

```
[ 0 -1]
```

Observation: With DBSCAN 2 clusters got created.

In [138]:

```python
from wordcloud import WordCloud, STOPWORDS
wordslist = df[df['labels'] == -1]['desc']
print(wordslist)
if(len(wordslist) > 0):
    stringlist = " ".join(wordslist)
    wordcloud = WordCloud(relative_scaling = 1.0,
                          stopwords = set(STOPWORDS)
                          ).generate(stringlist)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.title("cluster number " + str(-1))
    plt.show()
```

```
24
28      good beans not find grocery stores live ordere...
121
133     love water uplifting feel better drinking arse...
381         tasty gluten free option kids loved not spicy
456
601
642     baked first batch muffins pleased easy make ta...
644     product comes good company makes number excell...
648     bought local grocery store wish would read rev...
658     recently discovered sensitivities dairy wheat ...
718
792
820     like plockysthey like us plockys plockys mean ...
834     favorite gluten free dairy free flavored chips...
853     never met kettle brand chip not like chips gre...
903        salt free product purchased chips quite greasy
906     not need salt hide taste potato chips chips pr...
910     kettle brand chips crunchy would say regular p...
937     unless really really really like vinegar avoid...
968     salt vinegar chips favorite flavor think tried...
981     favorite home like ones sea salt also like bar...
1013    oh love chips hard find find usually per bag l...
1042
1057    price product certainly raises attention compa...
1127    licorice good taste daughter gluten free diet ...
1136
1183
1185    family favorite brand wheat free gluten free c...
1196    like green tea brews minutes max not stew tea ...
                              ...
3915    served gluten free bisquick pancake waffle mix...
3921    husband gluten intolerant gluten free products...
3924    product great pancakes baked goods blend bisqu...
3930    use gluten free bisquick make coated baked chi...
3945    received item time packaging great gluten free...
3947    no idea gluten free bisquick mix made pancakes...
3954    tried pamela second best bob red mill king art...
3955    found product local walmart trying vainly find...
3986    could not find gum grocery store ordered amazo...
4015    hot chocolate k cups not work like coffe k cup...
4024    wonderful dark chocolate flavor much deeper fl...
4042    sea salt easily administered salt mill salt sh...
4043    use buy sea salt instead regular salt like sea...
```

```
4151    girls love bars nice lunches quick snacks go g...
4345    feel like tried every hot cocoa k cups think b...
4360    really tasty hot chocolate like much dark choc...
4392    hot chocolate good right amount milk chocolate...
4395    hot chocolate tasted good whole family liked n...
4430    best hot cocoa tried keurig rich chocolate fla...
4459    not sweet best k cup hot chocolate tried thus ...
4478    absolute best hot cocoa keurig brewers hot coc...
4497    hot chocolate much better k cup hot chocolates...
4515    hot chocolate good flavorful compared hot choc...
4541    far best k cup hot cocoa tried nice rich flavo...
4569    really great hot chocolate price right usually...
4577    liked milk chocolate version tried good intens...
4586    grove square hot cocoa right amout milk chocol...
4591    excellent hot chocolate love love hot chocolat...
4607    really like selection choice three different h...
4844    good stuff hard find local pet stores ne pa on...
Name: desc, Length: 145, dtype: object
```

cluster number -1

Observation: Cluster -1 represents

In [139]:

```
wordslist = df[df['labels'] == 0]['desc']
print(wordslist)
if(len(wordslist) > 0):
    stringlist = " ".join(wordslist)
    wordcloud = WordCloud(relative_scaling = 1.0,
                          stopwords = set(STOPWORDS)
                          ).generate(stringlist)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.title("cluster number " + str(0))
    plt.show()
```

```
0       product available victor traps unreal course t...
1       used victor fly bait seasons ca not beat great...
2       received shipment could hardly wait try produc...
3       really good idea final product outstanding use...
4       glad cocker standard poodle puppy loves stuff ...
5       using food months find excellent fact two dogs...
6       nine cats crazy kibbles last thing want cat fo...
7       shipped day ordered arrived within days live o...
8       mix probably not something would want use ever...
9       description product disceptive product represe...
10      bought brand online indian grocery store usual...
11      use keep finicky toddler protein levels great ...
12      get busy home like sausage lot quick meal opti...
13      company american classic business years best h...
14      love pico pica adds flavor not hot eat least m...
15      thank goodness mexgrocer love pico pica sauce ...
16      different sauce nothing like anything find gro...
17      found product search edible gold leaf decided ...
18      purchased item cake called gold dust never tho...
19      used product multiple times fact purchased fou...
20      used super gold luster dust create exquisite c...
21      product allows make really big splashes provid...
22      cute affordable set son golf theme party great...
23      used one green ball etc golfer cake made one l...
25      golf set arrived quickly pictured birthday fiv...
26      natural ingredients no preservatives say fanta...
27      adzuki azuki beans ment used asian sweets make...
29      not good yummy smell like cloves cooking taste...
30      good stuff like lentils not need soak small fe...
31      sauce something permanent staple table everyth...
                              ...
4956    carabou mahogony worst tasting cup coffee ever...
4957    ordered mahogany caribou coffee k cups adore r...
4958    wife avid keurig coffee fans three brewers two...
4959    serious cup joe yummyness turned least friends...
4960    maybe greatest coffee ever made many time regu...
4961    surprised find taiwan shaped pineapple cakes c...
4962    really like pineapple shortcakes sold unlike m...
4963                                        started using
4964    using lourdes chimichurri years stuff awesome ...
4965    absolutely love product use chicken beef veggi...
4966    never sunchy malta drank lot goya available lo...
4967    item got house time surprised well package pac...
4968    since gluten free tried types gf breads expens...
4969    not good houston samba grill tasted great text...
```

```
4970    made recently holiday party never seen anythin...
4971    spent first five years life brazil american pa...
4972    love dont use much strong tastes great makes e...
4973    best olive oil not cooking olive oil work flav...
4974    nearby fresh easy neighborhood market stocks n...
4975                                get walmart gas station
4976    coffee really rich perfect morning ordered off...
4977    fresh great way get little chocolate life with...
4978    taste something like flax bread cornbread exce...
4979    wonderful dinner fresh fluke fried tempura bat...
4980    tried available discs kona blend one best regu...
4981    one best choices opinion also adore amazon nee...
4982    tried many tassimo flavors far favorite normal...
4983    bold blend great taste flavor comes bursting u...
4984    coffee available tassimo kona richest flavor f...
4985    coffee supposedly premium tastes watery thin n...
Name: desc, Length: 4841, dtype: object
```

cluster number 0

Observation: Cluster -1 : Bakery,cluster 1 : Categorized related to taster

[6] Conclusions

KMEAN Results

Method	No of samples	No of clusters
BOW	5000	25
TFIDF	5000	25
AVG W2VE	5000	15
TFIDF W2VE	50009	15

Agglomerative results

Method	No of samples	No of clusters
AVG W2VE	5000	3

Method	No of samples	No of clusters
TFIDF W2VE	50009	2

DBSCAN results

Method	No of samples	No of clusters
AVG W2VE	5000	2
TFIDF W2VE	50009	2