Amazon Fine Food Reviews Analysis

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

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

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

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

Attribute Information:

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

Objective:

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

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

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

[1]. Reading Data

[1.1] Loading the data

The dataset is available in two forms

1. .csv file
2. SQLite Database

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


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

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

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

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

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

from tqdm import tqdm
import os
```

```
C:\Users\hemant\AnacondaNew\lib\site-packages\smart_open\ssh.py:34: Use
rWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disable
d.  `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be d
isabled.  `pip install paramiko` to suppress')
C:\Users\hemant\AnacondaNew\lib\site-packages\gensim\utils.py:1197: Use
rWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_seria
l")
```

In [2]:
```
# using SQLite Table to read data.
con = sqlite3.connect(r'G:\database_assignment\Logistic_regression\data
base5.sqlite')
```

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

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

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

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

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

```
Number of data points in our data (525814, 10)
```

Out[2]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomin
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomin
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	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	COUNT(*)
0	#oc-R115TNMSPFT9I7	B005ZBZLT4	Breyton	1331510400	2	Overall its just OK when considering the price...	2

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)
1	#oc-R11D9D7SHXIJB9	B005HG9ESG	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B005ZBZLT4	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ESG	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBEV0	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	COUNT(*)
80638	AZY10LLTJ71NX	B001ATMQK2	undertheshrine "undertheshrine"	1296691200	5	I bought this 6 pack because for the price tha...	5

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

Out[6]:
```
393063
```

[2] Exploratory Data Analysis

[2.1] Data Cleaning: Deduplication

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

In [7]:

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

Out[7]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenon
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	

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenon
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 [8]:
```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=Tr
ue, inplace=False, kind='quicksort', na_position='last')
```

In [9]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time"
```

```
,"Text"}, keep='first', inplace=False)
final.shape
```

Out[9]: (364173, 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]: 69.25890143662969

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	HelpfulnessDenom
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	

In [12]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [13]:
```
#Before starting the next phase of preprocessing lets see the number of
 entries left
print(final.shape)

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

```
(364171, 10)
```

Out[13]:
```
1    307061
0     57110
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)
```

```
this witty little book makes my son laugh at loud. i recite it in the c
ar as we're driving along and he always can sing the refrain. he's lear
ned about whales, India, drooping roses:  i love all the new words this
book  introduces and the silliness of it all.  this is a classic book i
am  willing to bet my son will STILL be able to recite from memory when
he is  in college
==================================================
I was really looking forward to these pods based on the reviews.  Starb
ucks is good, but I prefer bolder taste.... imagine my surprise when I
ordered 2 boxes - both were expired! One expired back in 2005 for gosh
sakes.  I admit that Amazon agreed to credit me for cost plus part of s
hipping, but geez, 2 years expired!!!  I'm hoping to find local San Die
go area shoppe that carries pods so that I can try something different
than starbucks.
==================================================
Great ingredients although, chicken should have been 1st rather than ch
icken broth, the only thing I do not think belongs in it is Canola oil.
Canola or rapeseed is not someting a dog would ever find in nature and
if it did find rapeseed in nature and eat it, it would poison them. Tod
ay's Food industries have convinced the masses that Canola oil is a saf
e and even better oil than olive or virgin coconut, facts though say ot
herwise. Until the late 70's it was poisonous until they figured out a
```

```
herwise. Until the late 70's it was poisonous until they figured out a
way to fix that. I still like it but it could be better.
==================================================
Can't do sugar.  Have tried scores of SF Syrups.  NONE of them can touc
h the excellence of this product.<br /><br />Thick, delicious.  Perfec
t.  3 ingredients: Water, Maltitol, Natural Maple Flavor.  PERIOD.  No
chemicals.  No garbage.<br /><br />Have numerous friends & family membe
rs hooked on this stuff.  My husband & son, who do NOT like "sugar fre
e" prefer this over major label regular syrup.<br /><br />I use this as
my SWEETENER in baking: cheesecakes, white brownies, muffins, pumpkin p
ies, etc... Unbelievably delicious...<br /><br />Can you tell I like i
t? :)
==================================================
```

In [15]:

```python
# remove urls from text python: https://stackoverflow.com/a/40823105/40
84039
sent_0 = re.sub(r"http\S+", "", sent_0)
sent_1000 = re.sub(r"http\S+", "", sent_1000)
sent_150 = re.sub(r"http\S+", "", sent_1500)
sent_4900 = re.sub(r"http\S+", "", sent_4900)

print(sent_0)
```

```
this witty little book makes my son laugh at loud. i recite it in the c
ar as we're driving along and he always can sing the refrain. he's lear
ned about whales, India, drooping roses:  i love all the new words this
book  introduces and the silliness of it all.  this is a classic book i
am  willing to bet my son will STILL be able to recite from memory when
he is  in college
```

In [16]:

```python
# 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)
```

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

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

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

```
this witty little book makes my son laugh at loud. i recite it in the c
ar as we're driving along and he always can sing the refrain. he's lear
ned about whales, India, drooping roses:  i love all the new words this
book  introduces and the silliness of it all.  this is a classic book i
am  willing to bet my son will STILL be able to recite from memory when
he is  in college
==================================================
I was really looking forward to these pods based on the reviews.  Starb
ucks is good, but I prefer bolder taste.... imagine my surprise when I
ordered 2 boxes - both were expired! One expired back in 2005 for gosh
sakes.  I admit that Amazon agreed to credit me for cost plus part of s
hipping, but geez, 2 years expired!!!  I'm hoping to find local San Die
go area shoppe that carries pods so that I can try something different
than starbucks.
==================================================
Great ingredients although, chicken should have been 1st rather than ch
icken broth, the only thing I do not think belongs in it is Canola oil.
Canola or rapeseed is not someting a dog would ever find in nature and
if it did find rapeseed in nature and eat it, it would poison them. Tod
ay's Food industries have convinced the masses that Canola oil is a saf
e and even better oil than olive or virgin coconut, facts though say ot
herwise. Until the late 70's it was poisonous until they figured out a
way to fix that. I still like it but it could be better.
==================================================
Can't do sugar.  Have tried scores of SF Syrups.  NONE of them can touc
h the excellence of this product.Thick, delicious.  Perfect.  3 ingredi
```

```
ents: Water, Maltitol, Natural Maple Flavor.  PERIOD.  No chemicals.  N
o garbage.Have numerous friends & family members hooked on this stuff.
My husband & son, who do NOT like "sugar free" prefer this over major l
abel regular syrup.I use this as my SWEETENER in baking: cheesecakes, w
hite brownies, muffins, pumpkin pies, etc... Unbelievably delicious...C
an you tell I like it? :)
```

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]:

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

```
Great ingredients although, chicken should have been 1st rather than ch
icken broth, the only thing I do not think belongs in it is Canola oil.
Canola or rapeseed is not someting a dog would ever find in nature and
if it did find rapeseed in nature and eat it, it would poison them. Tod
ay is Food industries have convinced the masses that Canola oil is a sa
fe and even better oil than olive or virgin coconut, facts though say o
therwise. Until the late 70 is it was poisonous until they figured out
```

```
a way to fix that. I still like it but it could be better.
==================================================
```

In [19]:
```
#remove words with numbers python: https://stackoverflow.com/a/1808237
0/4084039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
this witty little book makes my son laugh at loud. i recite it in the c
ar as we're driving along and he always can sing the refrain. he's lear
ned about whales, India, drooping roses:  i love all the new words this
book  introduces and the silliness of it all.  this is a classic book i
am  willing to bet my son will STILL be able to recite from memory when
he is  in college
```

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

```
Great ingredients although chicken should have been 1st rather than chi
cken broth the only thing I do not think belongs in it is Canola oil Ca
nola or rapeseed is not someting a dog would ever find in nature and if
it did find rapeseed in nature and eat it it would poison them Today is
Food industries have convinced the masses that Canola oil is a safe and
even better oil than olive or virgin coconut facts though say otherwise
Until the late 70 is it was poisonous until they figured out a way to f
ix that I still like it but it could be better
```

In [21]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
# <br /><br /> ==> after the above steps, we are getting "br br"
# we are including them into stop words list
# instead of <br /> if we have <br/> these tags would have revmoved in
 the 1st step

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

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

In [22]:

```
# Combining all the above stundents
if not os.path.isfile('final.sqlite'):

    from tqdm import tqdm
    final_string=[]
    # 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.lo
wer() not in stopwords)
        final_string.append(sentance.strip())

         #############---- storing the data into .sqlite file ------###
#####################
    final['CleanedText']=final_string #adding a column of CleanedText w
hich displays the data after pre-processing of the review
    final['CleanedText']=final['CleanedText'].str.decode("utf-8")
        # store final table into an SQlLite table for future.
    conn = sqlite3.connect('final.sqlite')
    c=conn.cursor()
    conn.text_factory = str
    final05.to_sql('Reviews', conn,  schema=None, if_exists='replace',
 \
                 index=True, index_label=None, chunksize=None, dtype=No
ne)
    conn.close()
```

In [23]:
```
if os.path.isfile('final.sqlite'):
    conn = sqlite3.connect('final.sqlite')
    final1 = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score !=
 3 """, conn)
    conn.close()
else:
    print("Please the above cell")
```

In [24]:
```
final1.head(3)
final1['CleanedText'].head(5)
```

Out[24]:
```
0    witti littl book make son laugh loud recit car...
1    grew read sendak book watch realli rosi movi i...
2    fun way children learn month year learn poem t...
3    great littl book read nice rhythm well good re...
4    book poetri month year goe month cute littl po...
Name: CleanedText, dtype: object
```

[3.2] Preprocessing Review Summary

In [25]:
```
## Similartly you can do preprocessing for review summary also.
sorted_sample = final1.sort_values('Time', axis=0, ascending=True, inpl
ace=False, kind='quicksort', na_position='last')
sample_60000 = sorted_sample.iloc[0:100000]
```

[5] Assignment 11: Truncated SVD

Truncated-SVD

[5.1] Taking top features from TFIDF, SET 2

In [27]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer(ngram_range = (1,1) , max_features = 2000)
tfidf_train = tfidf_vect.fit_transform (sample_60000['CleanedText'])
```

In [28]:
```
top_2000 = tfidf_vect.get_feature_names()
```

[5.2] Calulation of Co-occurrence matrix

In [29]:
```
# Please write all the code with proper documentation

from tqdm import tqdm
n_neighbor = 5
occ_matrix_2000 = np.zeros((2000,2000))
for row in tqdm(sample_60000['CleanedText'].values):
    words_in_row = row.split()
    for index,word in enumerate(words_in_row):
        if word in top_2000:
```

```
                for j in range(max(index-n_neighbor,0),min(index+n_neighbor
,len(words_in_row)-1) + 1):
                    if words_in_row[j] in top_2000:
                        occ_matrix_2000[top_2000.index(word),top_2000.index
(words_in_row[j])] += 1
                    else:
                        pass
            else:
                pass
```

```
100%|██████████████████████████████████████████| 100000/100000 [42:50<00:00, 3
8.90it/s]
```

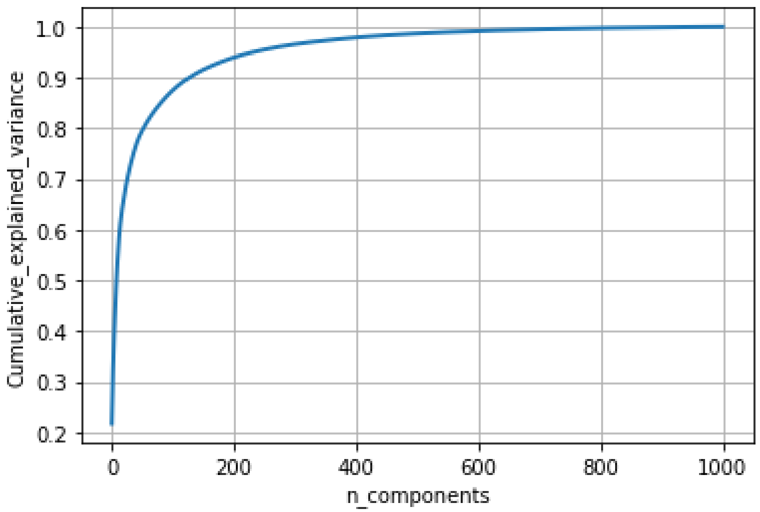
[5.3] Finding optimal value for number of components (n) to be retained.

In [31]:

```
# Please write all the code with proper documentation
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import StandardScaler
svd = TruncatedSVD(n_components = 1000)
svd_2000 = svd.fit_transform(occ_matrix_2000)

percentage_var_explained = svd.explained_variance_ / np.sum(svd.explain
ed_variance_);
cum_var_explained = np.cumsum(percentage_var_explained)
plt.figure(figsize=(6, 4))

plt.clf()
plt.plot(cum_var_explained, linewidth=2)
plt.axis('tight')
plt.grid()
plt.xlabel('n_components')
plt.ylabel('Cumulative_explained_variance')
plt.show()
```

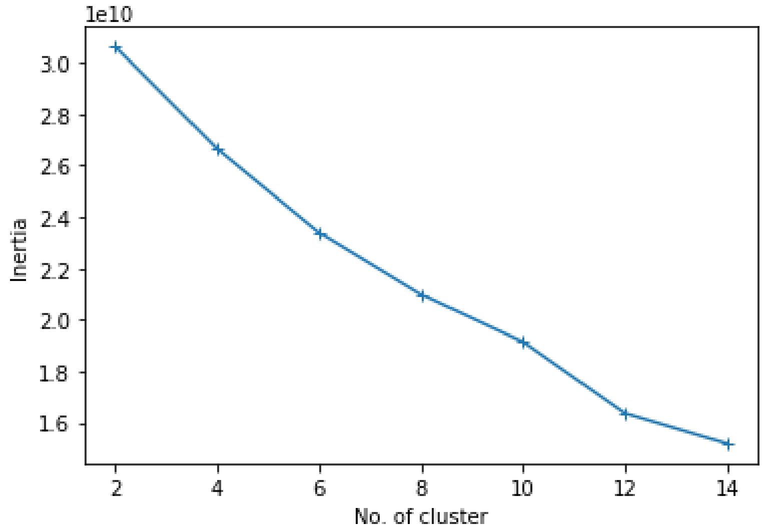


In [32]:
```
svd = TruncatedSVD(n_components = 150)
svd_2000 = svd.fit_transform(occ_matrix_2000)
```

[5.4] Applying k-means clustering

In [55]:
```
# Please write all the code with proper documentation

clusters = [2,4,6,8,10,12,14]
from sklearn.cluster import KMeans
dic = {}
for i in clusters:
    clus = KMeans(n_clusters = i)
    clus.fit(svd_2000)
    dic[i] = clus.inertia_
plt.plot(list(dic.keys()), list(dic.values()),'-+')
plt.xlabel("No. of cluster")
plt.ylabel("Inertia")
plt.show()
```



From above graph, we can observe that no.of cluster having 6 is optimal using elbow method.

In [72]:
```
optimal_k = KMeans(n_clusters = 6)
p = optimal_k.fit(svd_2000)
```

In [74]:
```
X1=sample_60000['CleanedText'].values
```

In [75]:
```
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []


for i in range(p.labels_.shape[0]):
    if p.labels_[i] == 0:
        cluster1.append(X1[i])
    elif p.labels_[i] == 1:
```

```
        cluster2.append(X1[i])
    elif p.labels_[i] == 2:
        cluster3.append(X1[i])
    elif p.labels_[i] == 3:
        cluster4.append(X1[i])
    elif p.labels_[i] == 4:
        cluster5.append(X1[i])
    elif p.labels_[i] == 5:
        cluster6.append(X1[i])
    elif p.labels_[i] == 6:
        cluster7.append(X1[i])


# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
print("\nNo. of reviews in Cluster-5 : ",len(cluster5))
print("\nNo. of reviews in Cluster-6 : ",len(cluster6))
```

```
No. of reviews in Cluster-1 :  1987

No. of reviews in Cluster-2 :  1

No. of reviews in Cluster-3 :  1

No. of reviews in Cluster-4 :  9

No. of reviews in Cluster-5 :  1

No. of reviews in Cluster-6 :  1
```

[5.5] Wordclouds of clusters obtained in the above section

In [76]:

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)
```

```python
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')


    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()
```

In [77]:

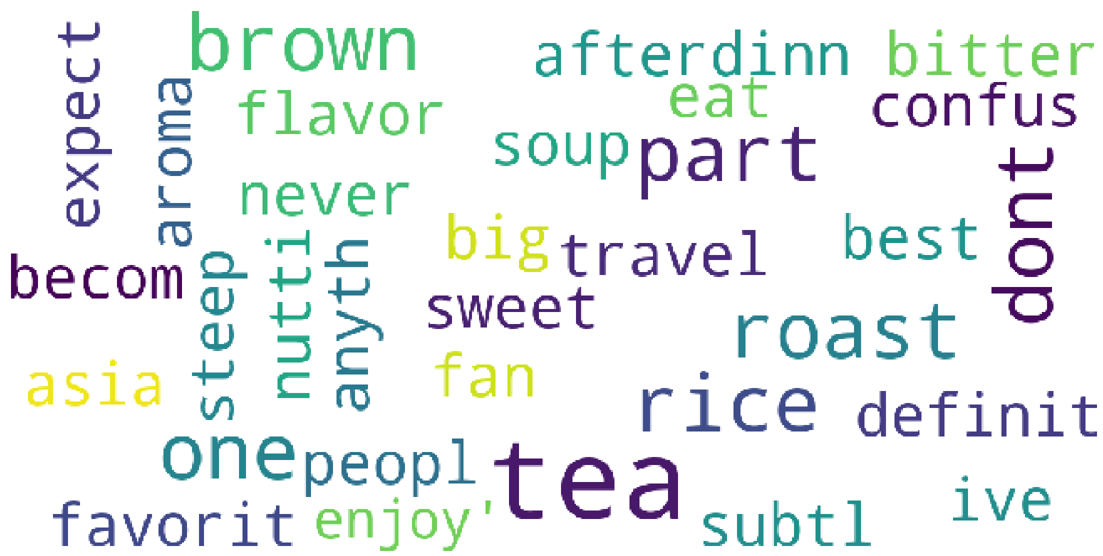
```python
show_wordcloud(cluster1)
show_wordcloud(cluster2)
```

blend fresh amazon right hard small nice come alway price give
got love cat drink tast look time
call cup wonder need water famili sugar packag movi
want chocol best work well bag howev day
mix see year enjoy two realli box
natur bar take treat
may coffe flavor candi know
still order dont delici make tri
contain good brand ive never
pod one food add better
way sauc sweet great
peopl actual perfect milk bottl first even
eat sinc product
chees use found excel friend favorit
thing buy ingredi tea ship
hot say find much littl made think put
stuff without mani

favorit day thank custom job tree
sister deliv
guarante love
ask good new cant
realli healthi gift
grown hous luck
move product
fortun sent
high plant warm even
exact still servic
recommend

In [78]:

```
show_wordcloud(cluster3)
show_wordcloud(cluster4)
show_wordcloud(cluster5)
```




chocol
milk
dark
chocol
box
fresh
extend
delici
place
coffe
caramel
fill
love
tast
bottl
flavor
ganach
take
butter
cream
thank
wine
candi
varieti
hint
stand
order
true
favorit
time
special
one
use
pod
fun
seal
still
gadget
came
includ
stopper
jesus
creami
purchas
dream
great
may
extra
new
coupl
tri
definit
kid
belgium
leonida
cherri
day
life
parent
pound
gold
give
power
small
describ
restaur
vin
packag
creat
ship
join
holder
music
come
suggest
locat
ribbon
vanilla
lift
made
wrap
five
think
antoinett
high
back
sweet
click
drink
someth
let
pump
fast
merlot
size
ton
done
world
even
backyard
inexpens
air
put
byo
incred
parti
snob
assort
never
aspir
insert
loos
coat
cup
hear
vacu
delic
along
indulg
rubber
bob
lord
birthday
four
safe
tool
home
simpli
paid
tea
buy
saver
knew
embodi
rest
friend
fall
find
know
break
went
protect
uniqu
adventur
sever
set
dress
extrem
potato


organ
cornstarch
one
qualiti
per
total
write
nutrit
theyv
bernieo
sodium
milk
cholesterol
enzym
fat
natur
individu
five
heat
amount
berni
pleas
turn
perhap
requir
protein
let
macaroni
serv
past
experi
blend
vitamin
daili
addit
two
microwav
carbohydr
cheddar
product
acid
evapor
semolina
mac
diet
specif
iron
help
salt
fiber
recommend
none
good
made
stovetop
late
tast
chees
citric
satur
inform
calcium
job
pure
mix
onion
sea
sugar
gram
question
paprika
purist
powder
name
comment
contain
big
cane
etc
tomato
juic
howev
garlic
pun
done
calori
cultur
cup
anni
flavor
bunni
mimim
fair
featur

In [79]:
```
show_wordcloud(cluster6)
```

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

In [80]:
```
# Please write all the code with proper documentation
from sklearn.metrics.pairwise import cosine_similarity
def similar_word_10(word):
    similarity = cosine_similarity(occ_matrix_2000)
    word_vect = similarity[top_2000.index(word)]
    print("Similar Word to",word)
    index = word_vect.argsort()[::-1][1:11]
    for j in range(len(index)):
        print((j+1),"Word",top_2000[index[j]] ,"is similar to",word,"\n
")
```

In [81]:
```
similar_word_10(top_2000[6])
```

```
Similar Word to acid
1 Word fatti is similar to acid

2 Word flavor is similar to acid

3 Word complex is similar to acid

4 Word coffe is similar to acid

5 Word bitter is similar to acid

6 Word vitamin is similar to acid

7 Word smooth is similar to acid

8 Word essenti is similar to acid

9 Word robust is similar to acid

10 Word tast is similar to acid
```

[6] Conclusions

We have taken top 2000 features based on idf values. Constructed a Co-occurance Matrix with help of these 2000 features Then applied Truncated SVD on co-occurance matrix with optimal no. of components. Kmeans on truncated SVD to analyse the clusters. Plotted the Word Cloud having cluster=8 to analyse what type of words it contain.

In [ ]: