Applying k-means,Agglomerative,DBSCAN Clustering 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)

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\HIMANSHU NEGI\Anaconda3\lib\site-packages\gensim\utils.py:1212: Use
rWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [2]:

```
# using SQLite Table to read data.
con = sqlite3.connect('database.sqlite')

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

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

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

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

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

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

Out[2]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenom
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
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	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B007Y59HVM	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ET0	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [5]:

```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[5]:

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

In [6]:

```
display['COUNT(*)'].sum()
```

Out[6]:

```
393063
```

[2] Exploratory Data Analysis

[2.1] Data Cleaning: Deduplication

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

In [7]:

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

Out[7]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDen
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 [8]:

```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=False, k
```

In [9]:

```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='firs
final.shape
```

Out[9]:

```
(46072, 10)
```

In [10]:

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

Out[10]:

```
92.144
```

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

In [11]:

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

display.head()
```

Out[11]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenc
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()
```

```
(46071, 10)
```

Out[13]:

```
1    38479
0     7592
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 [17]:

```
# 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)
```

```
My dogs loves this chicken but its a product from China, so we wont be buyin
g it anymore.  Its very hard to find any chicken products made in the USA bu
t they are out there, but this one isnt.  Its too bad too because its a good
product but I wont take any chances till they know what is going on with the
china imports.
==================================================
this is yummy, easy and unusual. it makes a quick, delicous pie, crisp or co
bbler. home made is better, but a heck of a lot more work. this is great to
have on hand for last minute dessert needs where you really want to impress
wih your creativity in cooking! recommended.
==================================================
Great flavor, low in calories, high in nutrients, high in protein! Usually p
rotein powders are high priced and high in calories, this one is a great bar
gain and tastes great, I highly recommend for the lady gym rats, probably no
t "macho" enough for guys since it is soy based...
==================================================
For those of you wanting a high-quality, yet affordable green tea, you shoul
d definitely give this one a try. Let me first start by saying that everyone
is looking for something different for their ideal tea, and I will attempt t
o briefly highlight what makes this tea attractive to a wide range of tea dr
inkers (whether you are a beginner or long-time tea enthusiast).  I have gon
e through over 12 boxes of this tea myself, and highly recommend it for the
following reasons:<br /><br />-Quality:  First, this tea offers a smooth qua
lity without any harsh or bitter after tones, which often turns people off f
rom many green teas.  I've found my ideal brewing time to be between 3-5 min
utes, giving you a light but flavorful cup of tea.  However, if you get dist
racted or forget about your tea and leave it brewing for 20+ minutes like I
sometimes do, the quality of this tea is such that you still get a smooth bu
t deeper flavor without the bad after taste.  The leaves themselves are whol
e leaves (not powdered stems, branches, etc commonly found in other brands),
and the high-quality nylon bags also include chunks of tropical fruit and ot
her discernible ingredients.  This isn't your standard cheap paper bag with
a mix of unknown ingredients that have been ground down to a fine powder, le
aving you to wonder what it is you are actually drinking.<br /><br />-Taste:
This tea offers notes of real pineapple and other hints of tropical fruits,
yet isn't sweet or artificially flavored.  You have the foundation of a high
-quality young hyson green tea for those true "tea flavor" lovers, yet the s
ubtle hints of fruit make this a truly unique tea that I believe most will e
njoy.  If you want it sweet, you can add sugar, splenda, etc but this really
is not necessary as this tea offers an inherent warmth of flavor through i
t's ingredients.<br /><br />-Price:  This tea offers an excellent product at
an exceptional price (especially when purchased at the prices Amazon offer
```

```
s).  Compared to other brands which I believe to be of similar quality (Migh
ty Leaf, Rishi, Two Leaves, etc.), Revolution offers a superior product at a
n outstanding price.  I have been purchasing this through Amazon for less pe
r box than I would be paying at my local grocery store for Lipton, etc.<br /
><br />Overall, this is a wonderful tea that is comparable, and even better
than, other teas that are priced much higher.  It offers a well-balanced cup
of green tea that I believe many will enjoy.  In terms of taste, quality, an
d price, I would argue you won't find a better combination that that offered
by Revolution's Tropical Green Tea.
==================================================
```

In [18]:

```
# 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)
```

```
My dogs loves this chicken but its a product from China, so we wont be buyin
g it anymore.  Its very hard to find any chicken products made in the USA bu
t they are out there, but this one isnt.  Its too bad too because its a good
product but I wont take any chances till they know what is going on with the
china imports.
```

In [19]:

```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-remove-all-tags-
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)
```

```
My dogs loves this chicken but its a product from China, so we wont be buyin
g it anymore.  Its very hard to find any chicken products made in the USA bu
t they are out there, but this one isnt.  Its too bad too because its a good
product but I wont take any chances till they know what is going on with the
china imports.
==================================================
this is yummy, easy and unusual. it makes a quick, delicous pie, crisp or co
bbler. home made is better, but a heck of a lot more work. this is great to
have on hand for last minute dessert needs where you really want to impress
wih your creativity in cooking! recommended.
==================================================
Great flavor, low in calories, high in nutrients, high in protein! Usually p
rotein powders are high priced and high in calories, this one is a great bar
gain and tastes great, I highly recommend for the lady gym rats, probably no
t "macho" enough for guys since it is soy based...
==================================================
For those of you wanting a high-quality, yet affordable green tea, you shoul
d definitely give this one a try. Let me first start by saying that everyone
is looking for something different for their ideal tea, and I will attempt t
o briefly highlight what makes this tea attractive to a wide range of tea dr
inkers (whether you are a beginner or long-time tea enthusiast).  I have gon
e through over 12 boxes of this tea myself, and highly recommend it for the
following reasons:-Quality:  First, this tea offers a smooth quality without
any harsh or bitter after tones, which often turns people off from many gree
n teas.  I've found my ideal brewing time to be between 3-5 minutes, giving
you a light but flavorful cup of tea.  However, if you get distracted or for
get about your tea and leave it brewing for 20+ minutes like I sometimes do,
the quality of this tea is such that you still get a smooth but deeper flavo
r without the bad after taste.  The leaves themselves are whole leaves (not
powdered stems, branches, etc commonly found in other brands), and the high-
quality nylon bags also include chunks of tropical fruit and other discernib
le ingredients.  This isn't your standard cheap paper bag with a mix of unkn
own ingredients that have been ground down to a fine powder, leaving you to
wonder what it is you are actually drinking.-Taste:  This tea offers notes o
f real pineapple and other hints of tropical fruits, yet isn't sweet or arti
ficially flavored.  You have the foundation of a high-quality young hyson gr
een tea for those true "tea flavor" lovers, yet the subtle hints of fruit ma
```

```
ke this a truly unique tea that I believe most will enjoy.  If you want it s
weet, you can add sugar, splenda, etc but this really is not necessary as th
is tea offers an inherent warmth of flavor through it's ingredients.-Price:
This tea offers an excellent product at an exceptional price (especially whe
n purchased at the prices Amazon offers).  Compared to other brands which I
believe to be of similar quality (Mighty Leaf, Rishi, Two Leaves, etc.), Rev
olution offers a superior product at an outstanding price.  I have been purc
hasing this through Amazon for less per box than I would be paying at my loc
al grocery store for Lipton, etc.Overall, this is a wonderful tea that is co
mparable, and even better than, other teas that are priced much higher.  It
offers a well-balanced cup of green tea that I believe many will enjoy.  In
terms of taste, quality, and price, I would argue you won't find a better co
mbination that that offered by Revolution's Tropical Green Tea.
```

In [20]:

```
# 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 [21]:

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

```
Great flavor, low in calories, high in nutrients, high in protein! Usually p
rotein powders are high priced and high in calories, this one is a great bar
gain and tastes great, I highly recommend for the lady gym rats, probably no
t "macho" enough for guys since it is soy based...
==================================================
```

In [22]:

```
#remove words with numbers python: https://stackoverflow.com/a/18082370/4084039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
My dogs loves this chicken but its a product from China, so we wont be buyin
g it anymore.  Its very hard to find any chicken products made in the USA bu
t they are out there, but this one isnt.  Its too bad too because its a good
product but I wont take any chances till they know what is going on with the
china imports.
```

In [23]:

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

```
Great flavor low in calories high in nutrients high in protein Usually prote
in powders are high priced and high in calories this one is a great bargain
and tastes great I highly recommend for the lady gym rats probably not macho
enough for guys since it is soy based
```

In [24]:

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

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', '
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"])
```

In [22]:

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

```
100%|████████████████████████████████████████████████████████████████████
███| 46071/46071 [00:28<00:00, 1880.33it/s]
```

In [26]:

```
preprocessed_reviews[1500]
```

Out[26]:

```
'great flavor low calories high nutrients high protein usually protein powde
rs high priced high calories one great bargain tastes great highly recommend
lady gym rats probably not macho enough guys since soy based'
```

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

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

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

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

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

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

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

[5.1] K-Means Clustering

In [23]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall
from sklearn.model_selection import GridSearchCV
from sklearn.cluster import KMeans
from sklearn.metrics import roc_curve,auc


from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_validate
from sklearn import tree
import pydotplus
from IPython.display import Image
from IPython.display import SVG
from graphviz import Source
from IPython.display import display
```

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

In [99]:

```
# Please write all the code with proper documentation
#code for VECTORIZER
count_vect = CountVectorizer(min_df = 10)
Xbow_tr = count_vect.fit_transform(preprocessed_reviews)

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

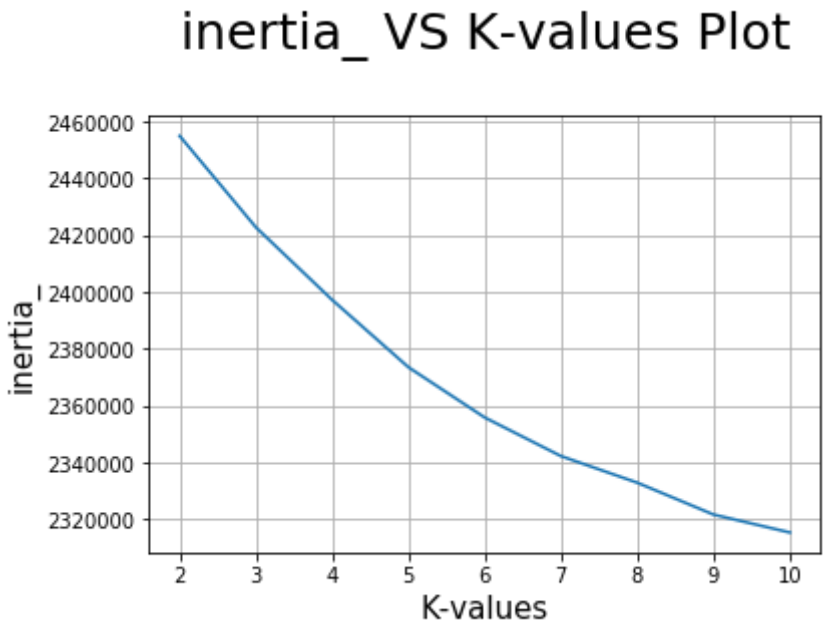
```
the type of count vectorizer : <class 'list'>
the shape of out text BOW vectorizer :  (46071, 8314)
the number of unique words : 8314
```

In [100]:

```
hyper = [2,3,4,5,6,7,8,9,10]
loss = []
for i in hyper:
    model = KMeans(n_clusters=i).fit(Xbow_tr)
    loss.append(model.inertia_)
```

In [101]:

```
# Draw Loss VS K values plot
plt.plot(hyper, loss)
plt.xlabel('K-values',size=15)
plt.ylabel('inertia_',size=15)
plt.title('inertia_ VS K-values Plot\n',size=25)
plt.grid()
plt.show()
```



In [102]:

```
optimal_k = 6
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = KMeans(n_clusters=optimal_k, n_jobs=-1).fit(Xbow_tr)
```

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

In [104]:

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

for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif kmeans.labels_[i] == 3:
        cluster4.append(reviews[i])
    elif kmeans.labels_[i] == 4:
        cluster5.append(reviews[i])
    else :
        cluster6.append(reviews[i])
```

In [106]:

```
# 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 :  10758

No. of reviews in Cluster-2 :  30146

No. of reviews in Cluster-3 :  1731

No. of reviews in Cluster-4 :  913

No. of reviews in Cluster-5 :  1158

No. of reviews in Cluster-6 :  1365
```

In [110]:

```
#code snippet from https://stackoverflow.com/questions/16645799/how-to-create-a-word-cloud-
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster1)
```

OBSERVATION: cluster1

1: As we can see in above cluster words like SWEET,TASTE,DRINK,FLAVOR,FOOD,EAT,CHOCOLATE,SWEET etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like FIND,BOX,NEVER,GOT etc in our cluster.

In [111]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster2)
```

OBSERVATION: cluster2

1: As we can see in above cluster words like TASTE,DELICIOUS,EAT,FOOD,FLAVOR,COFFEE,DRINK,CUP,SUGER etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like ONE,NEED,ORDER,SEEM,ORDERED etc in our cluster.

In [112]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster3)
```

OBSERVATION: cluster3

1: As we can see in above cluster words like GOOD,PRODUCT,DARK,CHOCOLATE,DRINK,CUP,WATER etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like GOT,RIGHT,WORK,FIND etc in our cluster.

In [113]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster4)
```

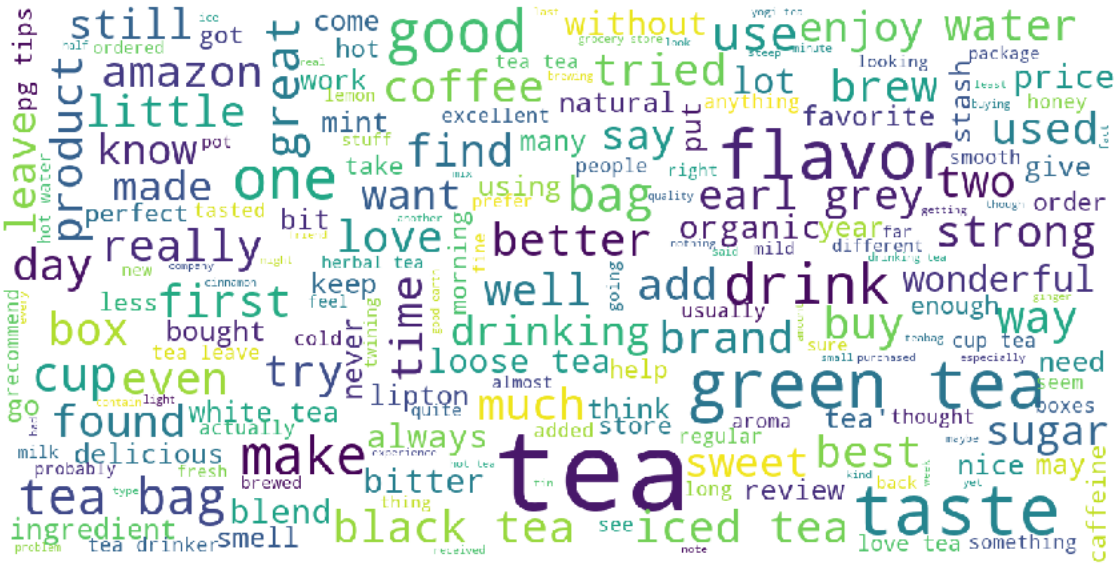
OBSERVATION: cluster4

1: As we can see in above cluster words like DOG,CAT,FOOD,EAT,CHICKEN,ANIMAL,SMELL,FEED etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like COME,REVIEW,FACT,AMOUNT etc in our cluster.

In [114]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster5)
```



OBSERVATION: cluster5

1: As we can see in above cluster words like TEA,GREEN,BLACK,COFFEE,DRINK,STRONG,LIPTONetc which shows reviews which are related are comes in one cluster.

2:There are some outliers like WHITE,NOTE,SEE,LONG etc in our cluster.

In [115]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster6)
```

OBSERVATION: cluster6

1: As we can see in above cluster words like WATER,COFFEE,DRINK,CUP,SMELL,BEAN etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like KIND,RICH,GOING,NOTHING etc in our cluster.

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

In [28]:

```
count_vect  = TfidfVectorizer(ngram_range=(1,2), min_df=10)
Xbow_tr = count_vect.fit_transform(preprocessed_reviews)

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

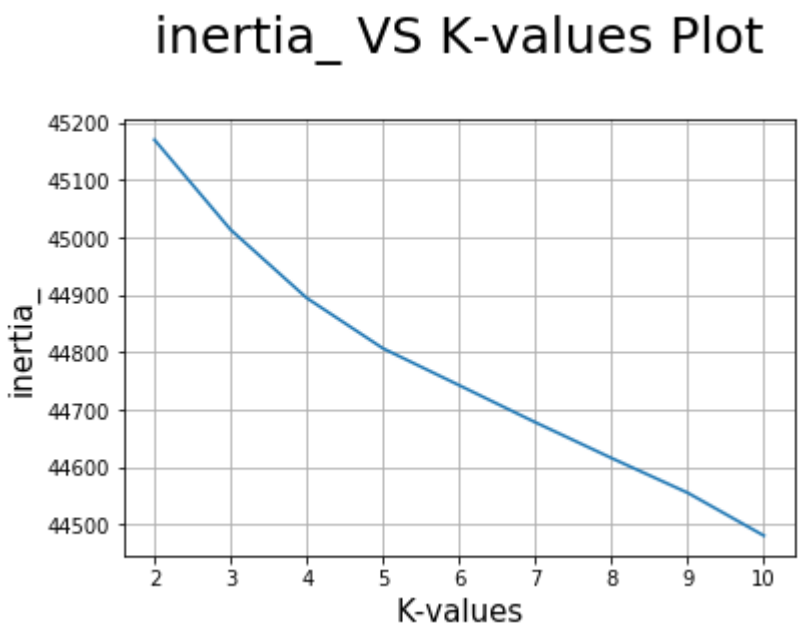
```
the type of count vectorizer : <class 'list'>
the shape of out text BOW vectorizer :  (46071, 27311)
the number of unique words : 27311
```

In [29]:

```
hyper = [2,3,4,5,6,7,8,9,10]
loss = []
for i in hyper:
    model = KMeans(n_clusters=i).fit(Xbow_tr)
    loss.append(model.inertia_)
```

In [30]:

```
# Draw Loss VS K values plot
plt.plot(hyper, loss)
plt.xlabel('K-values',size=15)
plt.ylabel('inertia_',size=15)
plt.title('inertia_ VS K-values Plot\n',size=25)
plt.grid()
plt.show()
```



In [31]:

```
optimal_k = 5
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = KMeans(n_clusters=optimal_k, n_jobs=-1).fit(Xbow_tr)
```

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

In [32]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif kmeans.labels_[i] == 3:
        cluster4.append(reviews[i])
    else :
        cluster5.append(reviews[i])
```

In [33]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  5401

No. of reviews in Cluster-2 :  1952

No. of reviews in Cluster-3 :  3844

No. of reviews in Cluster-4 :  31751

No. of reviews in Cluster-5 :  3123
```

In [34]:

```
#code snippet from https://stackoverflow.com/questions/16645799/how-to-create-a-word-cloud-
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster1)
```

OBSERVATION: cluster1

1: As we can see in above cluster words like DOG,CAT,EAT,FOOD,LOVE,BROUGHT,BUY etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like GREETING FORMULA COAT FACT BOX etc in our cluster.

In [35]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster2)
```

OBSERVATION: cluster2

1: As we can see in above cluster words like CHOCOLATE,COOKIE,SWEET,TASTE,DARK ,DELICIOUS,BAR,GOOD etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like BAG,RICH,GO,ORGANIC,FRIEND etc in our cluster.

In [36]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster3)
```

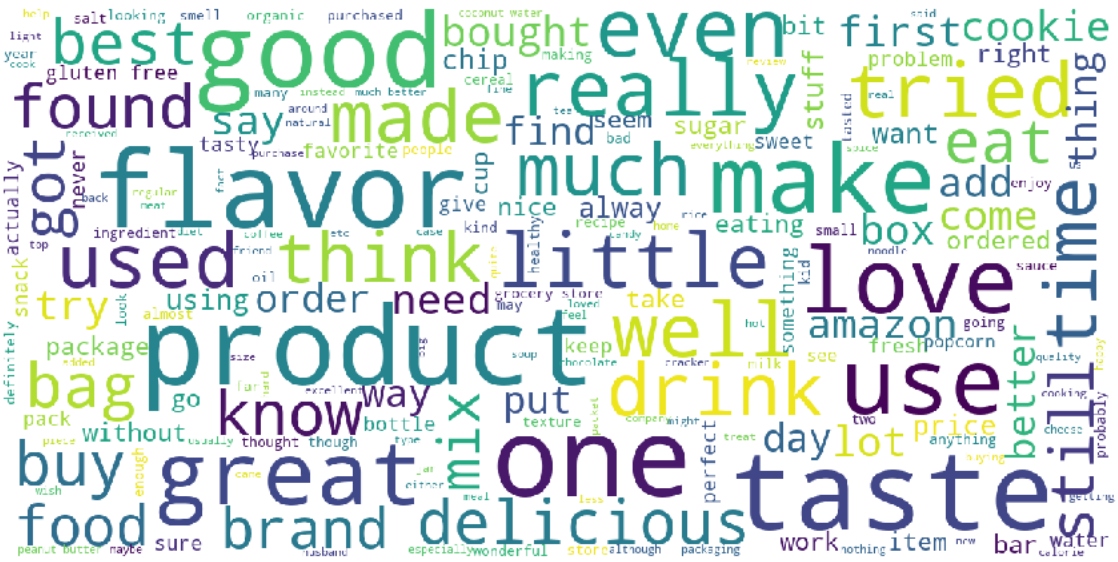
OBSERVATION: cluster3

1: As we can see in above cluster words like CUP,DRINK,COFFEE,CAFFINE,MILK,MORNING,DELICIOUS,STRONG,BEST etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like ROAST,ACID,GRIND,REVIEW etc in our cluster.

In [37]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster4)
```



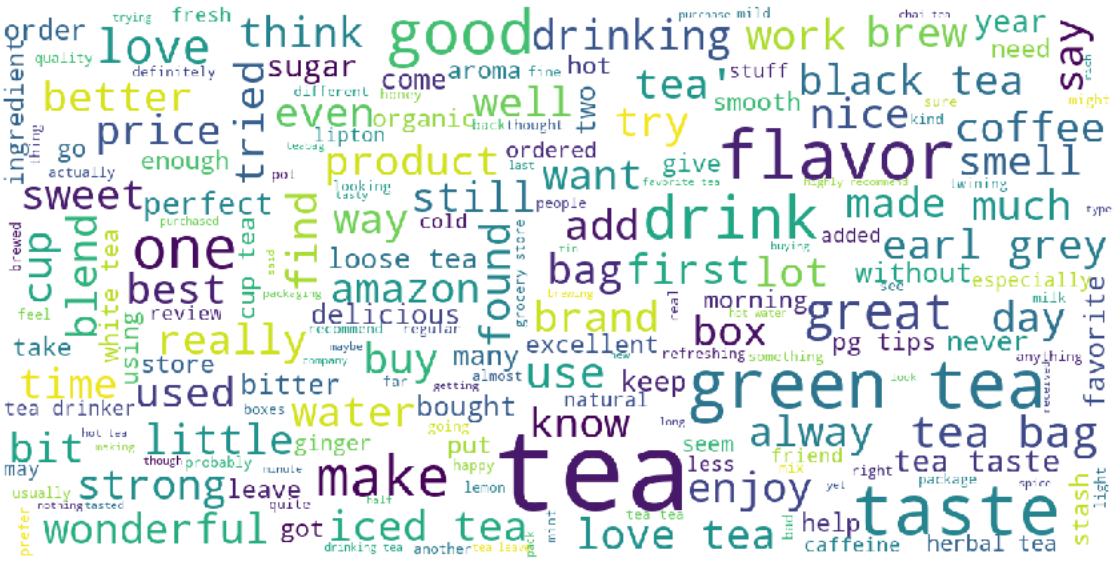
OBSERVATION: cluster4

1: As we can see in above cluster words like GOOD,BEST,BETTER,MADE,FLAVOR,TASTE,MAKE,NICE etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like HUSBAND,STORE,WORK,NOTHING etc in our cluster.

In [38]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster5)
```



OBSERVATION: cluster5

1: As we can see in above cluster words like TEA,STRONG,TASTE,GREEN,WATER,BRAND,FLAVOR,BAG, BEST,CUP,MORNING,SUGER,LIPTON,DELICIOUS etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like LEAVE,ANOTHER,THOUGH,LONG etc in our cluster.

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

In [41]:

```
# Please write all the code with proper documentation
# List of sentence in X_train text
sent_of_train=[]
for sent in preprocessed_reviews:
    sent_of_train.append(sent.split())


# Train your own Word2Vec model using your own train text corpus
# min_count = 5 considers only words that occured atleast 5 times
w2v_model=Word2Vec(sent_of_train,min_count=5,size=50, workers=4)

w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
```

```
number of words that occured minimum 5 times  12798
```

In [26]:

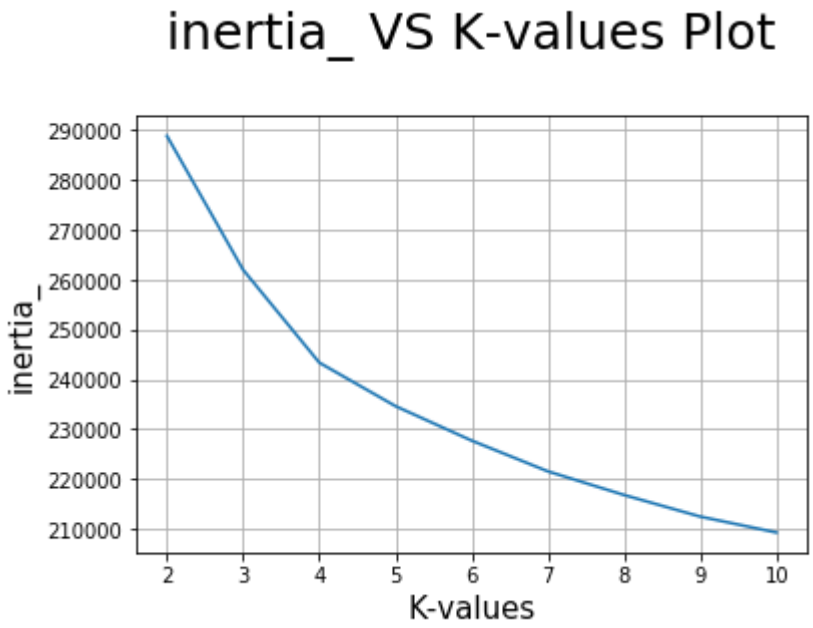
```
Xbow_tr = [];
for sent in sent_of_train:
    sent_vec = np.zeros(50)
    cnt_words =0;
    for word in sent: #
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    Xbow_tr.append(sent_vec)
```

In [30]:

```
hyper = [2,3,4,5,6,7,8,9,10]
loss = []
for i in hyper:
    model = KMeans(n_clusters=i).fit(Xbow_tr)
    loss.append(model.inertia_)
```

In [31]:

```
# Draw Loss VS K values plot
plt.plot(hyper, loss)
plt.xlabel('K-values',size=15)
plt.ylabel('inertia_',size=15)
plt.title('inertia_ VS K-values Plot\n',size=25)
plt.grid()
plt.show()
```



In [32]:

```
optimal_k = 5
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = KMeans(n_clusters=optimal_k, n_jobs=-1).fit(Xbow_tr)
```

In [33]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif kmeans.labels_[i] == 3:
        cluster4.append(reviews[i])
    else :
        cluster5.append(reviews[i])
```

In [34]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  11488

No. of reviews in Cluster-2 :  7547

No. of reviews in Cluster-3 :  6532

No. of reviews in Cluster-4 :  10082

No. of reviews in Cluster-5 :  10422
```

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

In [35]:

```
#code snippet from https://stackoverflow.com/questions/16645799/how-to-create-a-word-cloud-
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster1)
```

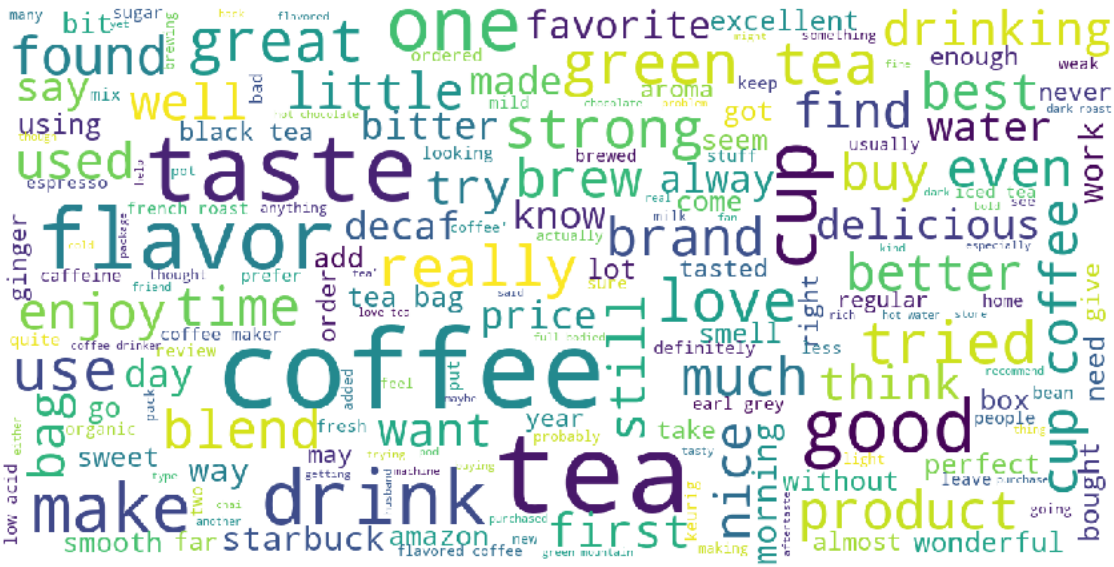
OBSERVATION: cluster1

1: As we can see in above cluster words like TASTE,DELICIOUS,EAT,FOOD, OIL,INGREDIENT,POPCORN,NOODLE etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like REVIEW,SOMETHING,QUITE,NEVER,GO etc in our cluster

In [36]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster2)
```



OBSERVATION: cluster2

1: As we can see in above cluster words like TEA,STRONG,TASTE,GREEN,WATER,BRAND,FLAVOR,BAG, BEST,CUP,MORNING,SUGER,AROMA,DELICIOUS etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like ORDER,MAY,FAR,AMAZON etc in our cluster.

In [37]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster3)
```

OBSERVATION: cluster3

1: As we can see in above cluster words like DOG,CAT,EAT,FOOD,LOVE,BROUGHT,BUY,DIET,BONE etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like MAY,HALF,GIVINGFIND,BABY etc in our cluster.

In [38]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster4)
```

OBSERVATION: cluster3

1: As we can see in above cluster words like DARK,CHOCOLATE,BAR,BUTTER,TASTE,DELICIOUS,CRUNCHY,CALORIE etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like PROBLEM,BAD,REGULAR,PAY,SEE etc in our cluster.

In [39]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster5)
```

OBSERVATION: cluster5

1: As we can see in above cluster words like BAG,COFFEE,TEA,TASTE,PRODUCT,STORE,GROCERY,PRICE etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like PEOPLE,NEW,PROBLEM,MANY,GO etc in our cluster.

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

In [40]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
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 [42]:

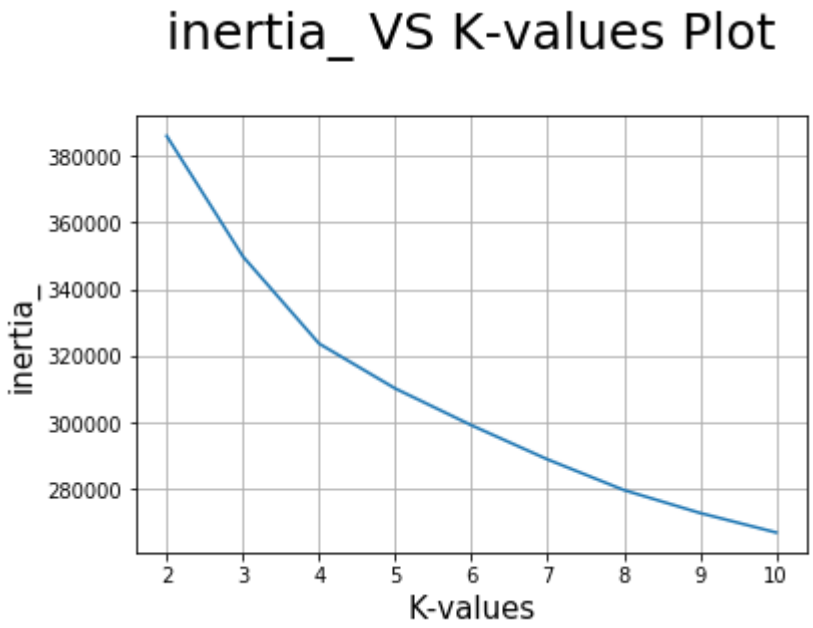
```
# TF-IDF weighted Word2Vec
tfidf_feat = model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfidf

Xbow_tr = []; # the tfidf-w2v for each sentence/review is stored in this list
row=0;
for sent in sent_of_train: # 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
    Xbow_tr.append(sent_vec)
    row += 1
```

In [43]:

```
hyper = [2,3,4,5,6,7,8,9,10]
loss = []
for i in hyper:
    model = KMeans(n_clusters=i).fit(Xbow_tr)
    loss.append(model.inertia_)
```

In [44]:

```
# Draw Loss VS K values plot
plt.plot(hyper, loss)
plt.xlabel('K-values',size=15)
plt.ylabel('inertia_',size=15)
plt.title('inertia_ VS K-values Plot\n',size=25)
plt.grid()
plt.show()
```



In [45]:

```
optimal_k = 4
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = KMeans(n_clusters=optimal_k, n_jobs=-1).fit(Xbow_tr)
```

In [46]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    else :
        cluster4.append(reviews[i])
```

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

In [47]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  6243

No. of reviews in Cluster-2 :  17938

No. of reviews in Cluster-3 :  6862

No. of reviews in Cluster-4 :  15028
```

In [48]:

```
#code snippet from https://stackoverflow.com/questions/16645799/how-to-create-a-word-cloud-
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster1)
```

OBSERVATION: cluster1

1: As we can see in above cluster words like DOG,CAT,BONE,FOOD,FLAVOR,EATING,CHEW,INGREDIENT etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like COME,SEE,FOUND,FAR,HOUR,BIG etc in our cluster.

In [49]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster2)
```

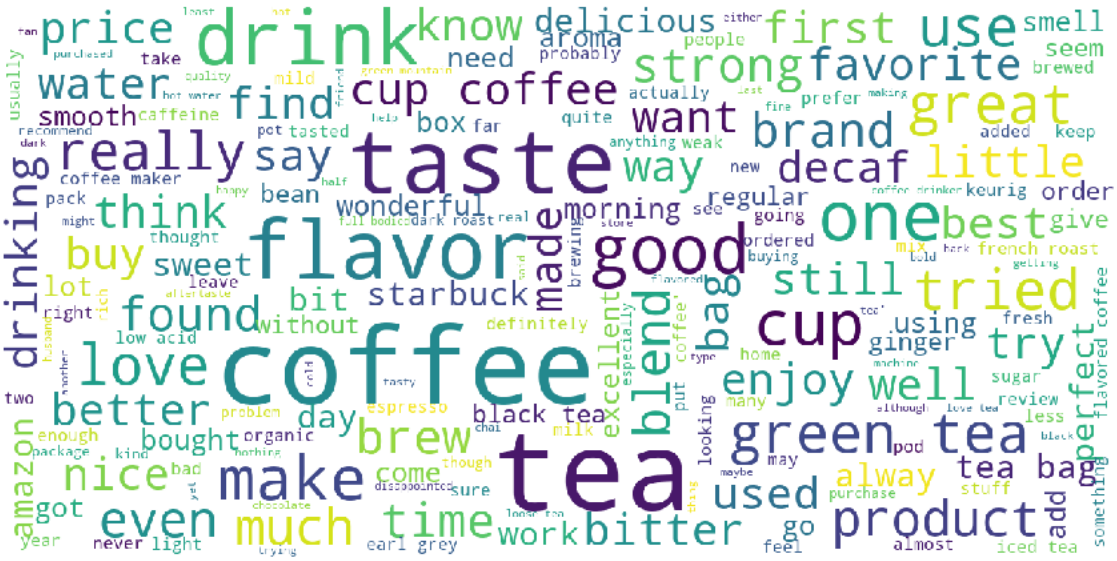
OBSERVATION: cluster2

1: As we can see in above cluster words like FLAVOR,TASTE,MIX,EAT,DRINK,DELICIOUS,CHOCOLATE,PEANUT,BUTTER etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like GO,FAR,SURE,MAY,KIND etc in our cluster.

In [50]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster3)
```



OBSERVATION: cluster3

1: As we can see in above cluster words like DRINK,CUP,COFFEE,TEA,FLAVOR,SWEET,BLACK,COLD etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like NEVER,TRYING,GO,MAYBE,BIT etc in our cluster.

In [51]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(cluster4)
```

OBSERVATION: cluster4

1: As we can see in above cluster words like TASTE,FLAVOR,POPCORN,FOOD,BAG,PURCHASE,PACKAGEING,GROCERY,STORE etc which shows reviews which are related are comes in one cluster.

2:There are some outliers like TIME,THOUGHT,WENT,BIG,MAYLAST etc in our cluster.

[5.2] Agglomerative Clustering

In [36]:

```
data_pos =final[final["Score"] == 1].sample(n = 2500)
data_neg = final[final["Score"] == 0].sample(n = 2500)
aggro = pd.concat([data_pos, data_neg])
aggro.shape
```

Out[36]:

```
(5000, 10)
```

In [37]:

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

```
100%|██████████████████████████████████████████████████████████████████████████████
██████| 5000/5000 [00:03<00:00, 1619.25it/s]
```

In [31]:

```
from sklearn.cluster import AgglomerativeClustering
```

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

In [38]:

```
# Please write all the code with proper documentation
# List of sentence in X_train text
sent_of_train=[]
for sent in preprocessed_reviews:
    sent_of_train.append(sent.split())


# Train your own Word2Vec model using your own train text corpus
# min_count = 5 considers only words that occured atleast 5 times
w2v_model=Word2Vec(sent_of_train,min_count=5,size=50, workers=4)

w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
```

```
number of words that occured minimum 5 times  4210
```

In [39]:

```
Xbow_tr = [];
for sent in sent_of_train:
    sent_vec = np.zeros(50)
    cnt_words =0;
    for word in sent: #
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    Xbow_tr.append(sent_vec)
```

In [40]:

```
optimal_k =2
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [41]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    else :
        cluster2.append(reviews[i])
```

In [42]:

```
# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
```

```
No. of reviews in Cluster-1 :  3826

No. of reviews in Cluster-2 :  1174
```

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

In [45]:

```
#code snippet from https://stackoverflow.com/questions/16645799/how-to-create-a-word-cloud-
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
```

CLUSTER 1

CLUSTER 2

OBSERVATION: cluster1,2

1: As we can see in above two clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

In [46]:

```
optimal_k =3
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [47]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    else :
        cluster3.append(reviews[i])
```

In [48]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  2567

No. of reviews in Cluster-2 :  1174

No. of reviews in Cluster-3 :  1259
```

In [49]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
print("CLUSTER 3")
show_wordcloud(cluster3)
```

CLUSTER 1

CLUSTER 2

CLUSTER 3

OBSERVATION: cluster1,2,3

1: As we can see in above three clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

In [50]:

```
optimal_k =4
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [51]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    else :
        cluster4.append(reviews[i])
```

In [52]:

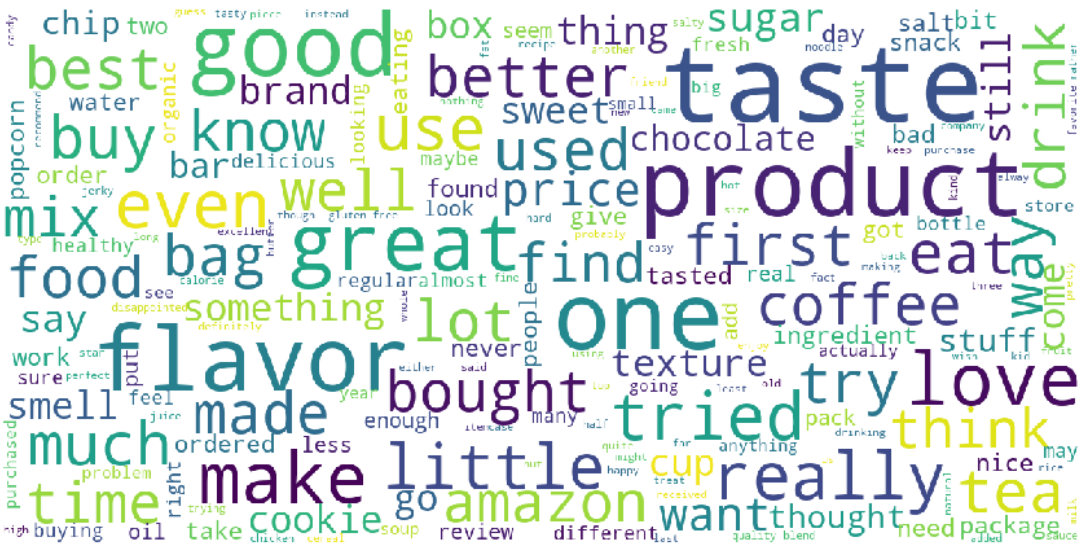
```
# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
```

```
No. of reviews in Cluster-1 :  1972

No. of reviews in Cluster-2 :  1174

No. of reviews in Cluster-3 :  1259

No. of reviews in Cluster-4 :  595
```

In [53]:

```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
print("CLUSTER 3")
show_wordcloud(cluster3)
print("CLUSTER 4")
show_wordcloud(cluster4)
```

CLUSTER 1



OBSERVATION: cluster1,2,3,4

1: As we can see in above four clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

In [54]:

```
optimal_k =5
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [55]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif kmeans.labels_[i] == 3:
        cluster4.append(reviews[i])
    else :
        cluster5.append(reviews[i])
```

In [56]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  1174

No. of reviews in Cluster-2 :  595

No. of reviews in Cluster-3 :  1259

No. of reviews in Cluster-4 :  1126

No. of reviews in Cluster-5 :  846
```

In [59]:

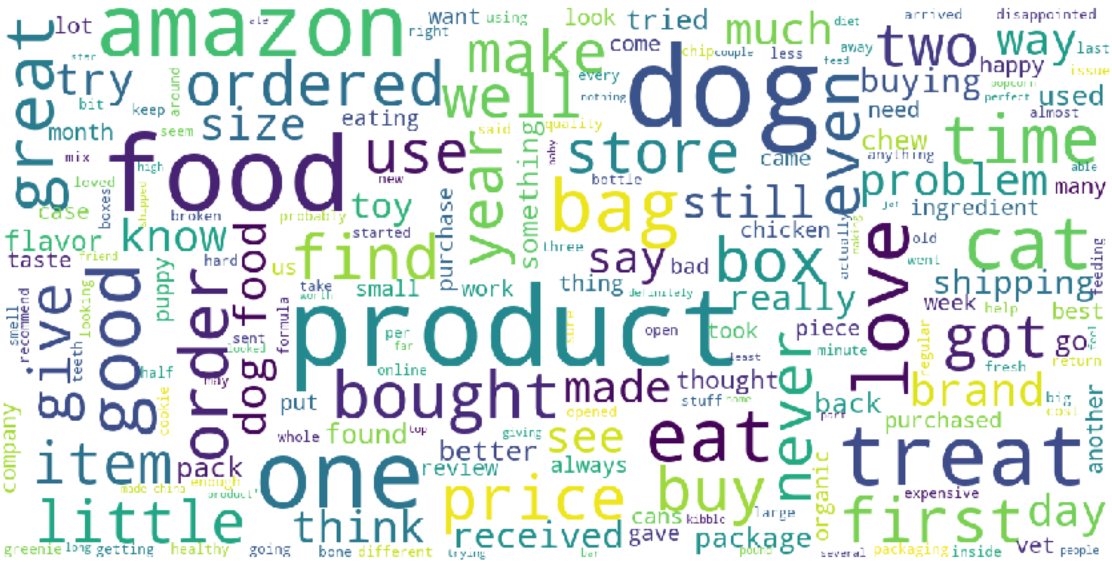
```
def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
print("CLUSTER 3")
show_wordcloud(cluster3)
```

CLUSTER 1

CLUSTER 2


amazon
product
taste
bought
treat
used
find
flavor
little
price
food
one
good
make
dog
love
made
first
well
use
day
bag
buy
box
really
even
time
great
much
know
way
ingredient
coffee
tea
brand
received
better
think
say
take
order
still
try
less
keep
sugar
sweet
quality
company
water
chocolate
thing
without
item
never
need
back
purchased
found
give
got
two
package
health
bottle
new
small
put
tried
seed
ordered
review
problem
cup
shipping
something
work

CLUSTER 3


amazon
dog
food
product
one
great
good
order
give
little
item
bought
price
buy
eat
treat
first
love
never
even
cat
box
bag
year
time
store
make
well
ordered
size
two
way
much
tried
problem
still
find
know
made
got
brand
really
shipping
think
received
package
dog food
toy
use
say
flavor
taste
try
lot
puppy
ingredient
chicken
back
see
better

In [60]:

```
print("CLUSTER 4")
show_wordcloud(cluster4)
print("CLUSTER 5")
show_wordcloud(cluster5)
```

CLUSTER 4

CLUSTER 5

OBSERVATION: cluster1,2,3,4,5

1: As we can see in above five clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

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

In [61]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
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 [62]:

```
# 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

Xbow_tr = []; # the tfidf-w2v for each sentence/review is stored in this list
row=0;
for sent in sent_of_train: # 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
    Xbow_tr.append(sent_vec)
    row += 1
```

In [64]:

```
optimal_k =2
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [65]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    else :
        cluster2.append(reviews[i])
```

In [66]:

```
# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
```

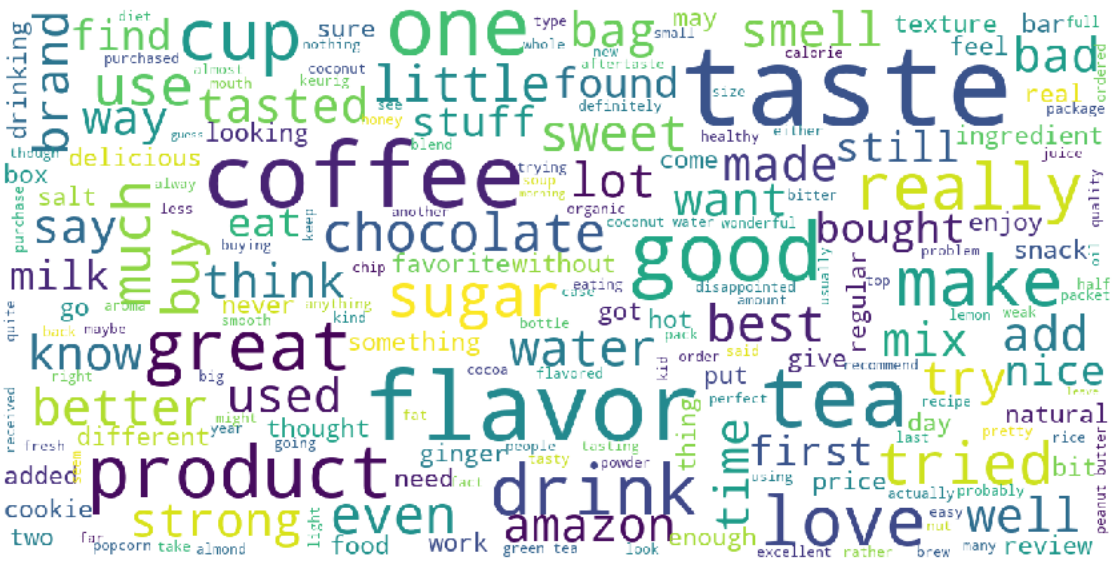
```
No. of reviews in Cluster-1 :  2811

No. of reviews in Cluster-2 :  2189
```

In [67]:

```
#code snippet from https://stackoverflow.com/questions/16645799/how-to-create-a-word-cloud-
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        #max_words=20,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
```

CLUSTER 1

CLUSTER 2



OBSERVATION: cluster1,2

1: As we can see in above two clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

In [68]:

```
optimal_k =3
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [69]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    else :
        cluster3.append(reviews[i])
```

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

In [70]:

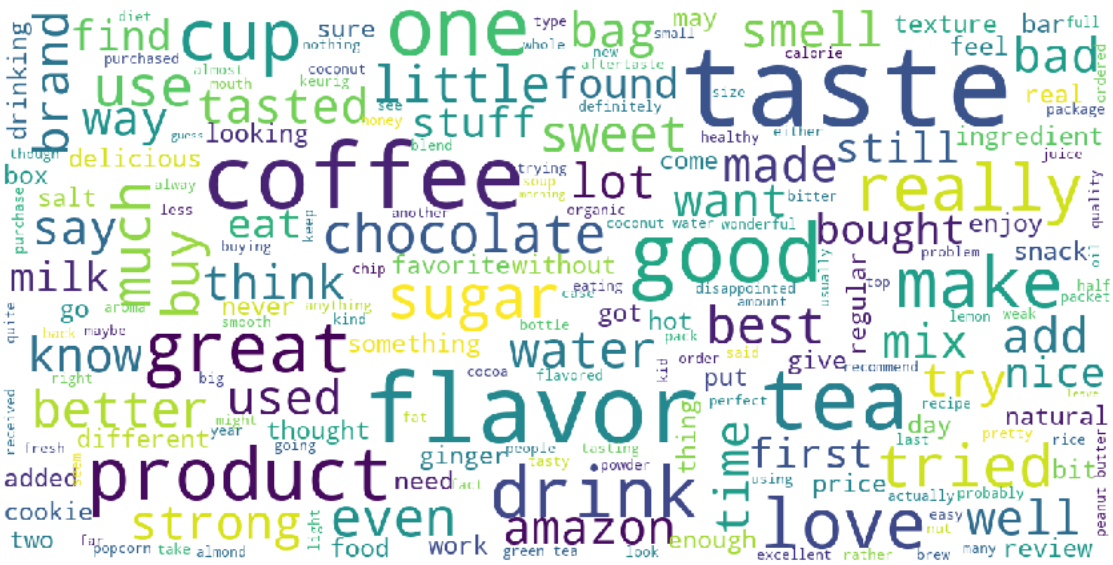
```
# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
```

```
No. of reviews in Cluster-1 :  2189

No. of reviews in Cluster-2 :  2058

No. of reviews in Cluster-3 :  753
```

In [71]:

```
print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
print("CLUSTER 3")
show_wordcloud(cluster3)
```

CLUSTER 1



CLUSTER 2

CLUSTER 3

OBSERVATION: cluster1,2,3

1: As we can see in above three clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

In [72]:

```
optimal_k =4
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [73]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    else :
        cluster4.append(reviews[i])
```

In [74]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  2058

No. of reviews in Cluster-2 :  1962

No. of reviews in Cluster-3 :  753

No. of reviews in Cluster-4 :  227
```

In [75]:

```
print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
print("CLUSTER 3")
show_wordcloud(cluster3)
print("CLUSTER 4")
show_wordcloud(cluster4)
```

```
CLUSTER 1
```

OBSERVATION: cluster1,2,3,4

1: As we can see in above four clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

In [76]:

```
optimal_k =5
# Variable that will be used in the conclusion
bow_means_k = optimal_k

# Implementing K-Means++ using optimal value of K
kmeans = AgglomerativeClustering(n_clusters=optimal_k).fit(Xbow_tr)
```

In [77]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []


for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif kmeans.labels_[i] == 3:
        cluster4.append(reviews[i])
    else :
        cluster5.append(reviews[i])
```

In [78]:

```
# 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))
```

```
No. of reviews in Cluster-1 :  1962

No. of reviews in Cluster-2 :  734

No. of reviews in Cluster-3 :  753

No. of reviews in Cluster-4 :  227

No. of reviews in Cluster-5 :  1324
```

In [79]:

```
print("CLUSTER 1")
show_wordcloud(cluster1)
print("CLUSTER 2")
show_wordcloud(cluster2)
print("CLUSTER 3")
show_wordcloud(cluster3)
```

CLUSTER 1

CLUSTER 2

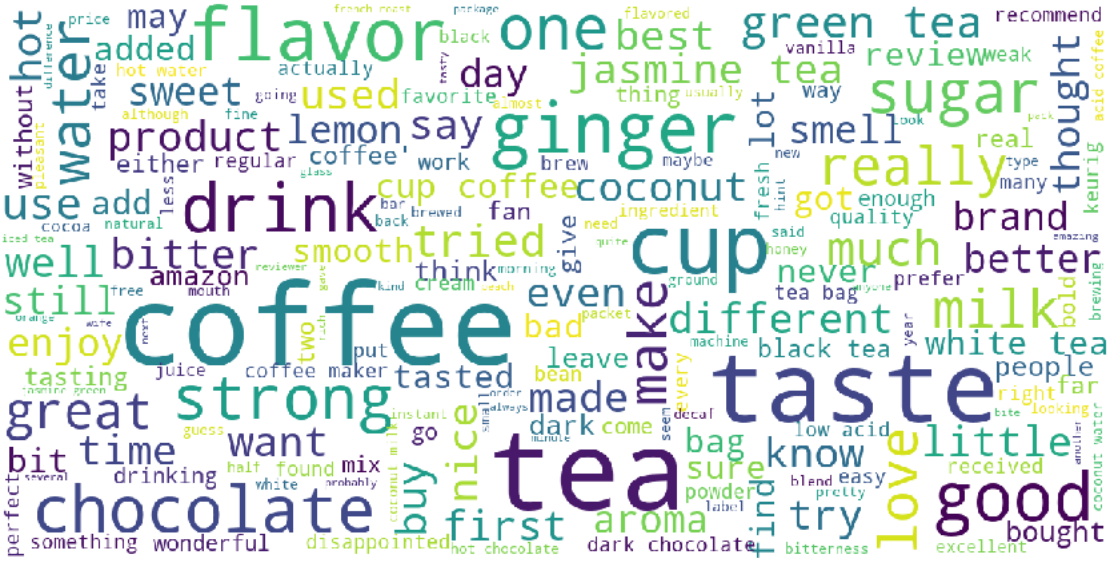
CLUSTER 3


purchase
flavor
even
think
brand
taste
box
sauce
bit
bad
add
work
ingredient
maybe
better
year
texture
sugar
day
one
find
really
using
amazon
buy
make
time
made
used
thing
eat
bought
need
say
come
company
ordered
review
favorite
quality
seem
several
container
small
product
back
water
still
less
package
tried
two
bag
price
coffee
little
best
sweet
real
far
tea
love
found
keep
good
take
got
may
mix
new
many
great
go
want
put
stuff
food
delicious
try
use
give
week
smell
oil
look
cup
fresh
enough
first
much
way
well
order
know
going
lot
something
pretty
different
received
trying

In [80]:

```
print("CLUSTER 4")
show_wordcloud(cluster4)
print("CLUSTER 5")
show_wordcloud(cluster5)
```

CLUSTER 4



CLUSTER 5

OBSERVATION: cluster1,2,3,4

1: As we can see in above four clusters many words are related to each other in same clusters shows that reviews which are related are in same cluster.

2:There are some words in cluster which also are in other cluster, some of them may be outliers.

[5.3] DBSCAN Clustering

In [81]:

```
from sklearn.cluster import DBSCAN
```

[5.3.1] Applying DBSCAN on AVG W2V, SET 3

In [125]:

```
# List of sentence in X_train text
sent_of_train=[]
for sent in preprocessed_reviews:
    sent_of_train.append(sent.split())

w2v_model=Word2Vec(sent_of_train,min_count=5,size=50, workers=4)

w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
```

```
number of words that occured minimum 5 times  4210
```

In [126]:

```
# List of sentence in X_train text
sent_of_train=[]
for sent in preprocessed_reviews:
    sent_of_train.append(sent.split())


# Train your own Word2Vec model using your own train text corpus
# min_count = 5 considers only words that occured atleast 5 times
w2v_model=Word2Vec(sent_of_train,min_count=5,size=50, workers=4)

w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
```

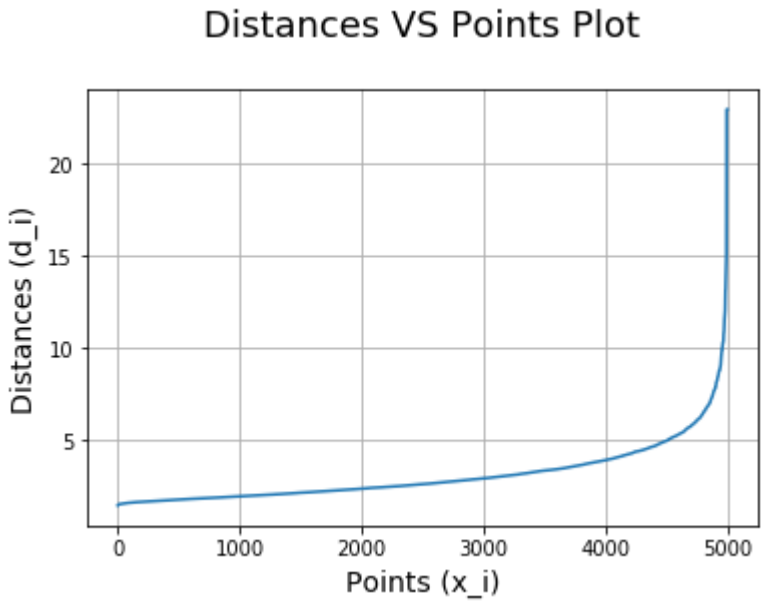
```
number of words that occured minimum 5 times  4210
```

In [127]:

```
# function to determinethe distance of nth-nearest neighbour to all points in a multi-dimen
def n_neighbour(vectors , n):
    distance = []
    for point in vectors:
        temp = np.sort(np.sum((vectors-point)**2,axis=1),axis=None)
        distance.append(temp[n])
    return np.sqrt(np.array(distance))
```

In [128]:

```
#Standardising the data
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler

data = StandardScaler().fit_transform(Xbow_tr)
min_points = 2*data.shape[1]

# Computing distances of nth-nearest neighbours
distances = n_neighbour(data,min_points)
sorted_distance = np.sort(distances)
points = [i for i in range(data.shape[0])]

# Draw distances(d_i) VS points(x_i) plot
plt.plot(points, sorted_distance)
plt.xlabel('Points (x_i)',size=14)
plt.ylabel('Distances (d_i)',size=14)
plt.title('Distances VS Points Plot\n',size=18)
plt.grid()
plt.show()
```



In [129]:

```
# Function definition for implementing DBSCAN
def dbscan(epsilon, samples, Data):
    from sklearn.cluster import DBSCAN
    db = DBSCAN(eps=epsilon, min_samples=samples, n_jobs=-1).fit(Data)

    # Number of clusters in labels, ignoring noise(-1) if present.
    n_clusters = len(set(db.labels_))
    print("Number of clusters for MinPts = %d and Epsilon = %f is : %d "%(samples,epsilon,r
    print("Labels(-1 is for Noise) : ",set(db.labels_))
    print()
    return db
```

In [130]:

```
optimal_eps = 5
# Clustering with right epsilon
db1 = dbscan(optimal_eps, min_points, data)


# Clustering with  epsilon =6
db2 = dbscan(6, min_points, data)


# Clustering with epsilon =7
db3 = dbscan(7, min_points, data)


# Clustering with epsilon =8
db4 = dbscan(8, min_points, data)
```

```
Number of clusters for MinPts = 100 and Epsilon = 5.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}

Number of clusters for MinPts = 100 and Epsilon = 6.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}

Number of clusters for MinPts = 100 and Epsilon = 7.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}

Number of clusters for MinPts = 100 and Epsilon = 8.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}
```

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

In [131]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster = []
for i in range(db1.labels_.shape[0]):
     cluster.append(reviews[i])
print("No. of reviews in Cluster-1 : ",len(cluster))
```

```
No. of reviews in Cluster-1 :  5000
```

In [132]:

```
print("CLUSTER")
show_wordcloud(cluster)
```

CLUSTER

OBSERVATION: cluster1

1: As we can see in above cluster words like FOOD,TASTE,EAT,FLAVOR,DRINK,SUGER,COOKIE etc which shows reviews which are very closly related are comes in one cluster.

2:There are some outliers like DAY,PUT,REVIEW,SURE etc in our cluster.

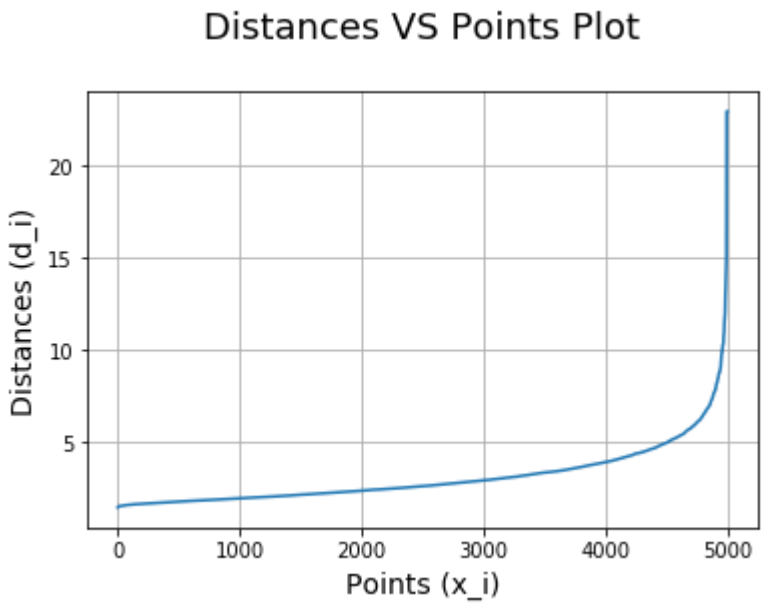
[5.3.3] Applying DBSCAN on TFIDF W2V, SET 4

In [92]:

```
# function to determinethe distance of nth-nearest neighbour to all points in a multi-dimen
def n_neighbour(vectors , n):
    distance = []
    for point in vectors:
        temp = np.sort(np.sum((vectors-point)**2,axis=1),axis=None)
        distance.append(temp[n])
    return np.sqrt(np.array(distance))
```

In [114]:

```
#Standardising the data
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler

data = StandardScaler().fit_transform(Xbow_tr)
min_points = 2*data.shape[1]

# Computing distances of nth-nearest neighbours
distances = n_neighbour(data,min_points)
sorted_distance = np.sort(distances)
points = [i for i in range(data.shape[0])]

# Draw distances(d_i) VS points(x_i) plot
plt.plot(points, sorted_distance)
plt.xlabel('Points (x_i)',size=14)
plt.ylabel('Distances (d_i)',size=14)
plt.title('Distances VS Points Plot\n',size=18)
plt.grid()
plt.show()
```



In [105]:

```
# Function definition for implementing DBSCAN
def dbscan(epsilon, samples, Data):
    from sklearn.cluster import DBSCAN
    db = DBSCAN(eps=epsilon, min_samples=samples, n_jobs=-1).fit(Data)

    # Number of clusters in labels, ignoring noise(-1) if present.
    n_clusters = len(set(db.labels_))
    print("Number of clusters for MinPts = %d and Epsilon = %f is : %d "%(samples,epsilon,r
    print("Labels(-1 is for Noise) : ",set(db.labels_))
    print()
    return db
```

In [107]:

```
optimal_eps = 5
# Clustering with right epsilon
db1 = dbscan(optimal_eps, min_points, data)


# Clustering with  epsilon =6
db2 = dbscan(6, min_points, data)


# Clustering with epsilon =7
db3 = dbscan(7, min_points, data)


# Clustering with epsilon =8
db4 = dbscan(8, min_points, data)
```

```
Number of clusters for MinPts = 100 and Epsilon = 5.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}

Number of clusters for MinPts = 100 and Epsilon = 6.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}

Number of clusters for MinPts = 100 and Epsilon = 7.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}

Number of clusters for MinPts = 100 and Epsilon = 8.000000 is : 2
Labels(-1 is for Noise) :  {0, -1}
```

In [123]:

```
reviews = preprocessed_reviews
# Getting all the reviews in different clusters
cluster = []
for i in range(db1.labels_.shape[0]):
     cluster.append(reviews[i])
print("No. of reviews in Cluster-1 : ",len(cluster))
```

```
No. of reviews in Cluster-1 :  5000
```

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

In [124]:

```
print("CLUSTER")
show_wordcloud(cluster)
```

CLUSTER

OBSERVATION: cluster1

1: As we can see in above cluster words like FOOD,TASTE,PRODUCT,TEA,CHOCOLATE,FLAVOR,DRINK,SUGER,POPCORN etc which shows reviews which are very closly related are comes in one cluster.

2:There are some outliers OLD,DAY,LOOK,COME etc in our cluster.

[6] Conclusions

NOTE--->>> WE HAVE TAKEN 5000 POINTS FOR Agglomerative AND DBSCAN CLUSTERING AND 50k for k-means CLUSTERING.

Procedure Followed :

STEP 1 :- Text Preprocessing.

STEP 2 :- Taking all text data and ignoring class variable .

STEP 3:- Training the vectorizer on text_data and later applying same vectorizer on text_data to transform it into vectors

STEP 4:- Standardizing the vectorized data

STEP 5:- Applying the Elbow Method in order to find the right value of k and Epsilon for k-means and DBSCAN clustering algorithm.

STEP 6:- Draw K VS inertia_ for k-means and distances VS points plot for DBSCAN.

STEP 7:- Implementing k-means with optimal k and DBSCAN with various values of Epsilon including the optimal value of Epsilon.

STEP 8:- Draw Wordclouds of clusters for k-means,Agglomerative,DBSCAN with optimal k and Epsilon values.

Repeat from STEP 3 to STEP 8 for each of these four vectorizers : Bag Of Words(BoW), TFIDF, Avg Word2Vec and TFIDF Word2Vec only for k-means,for rest of two we done it on Word2Vec and TFIDF Word2Vec only.

---> Implementing Agglomerative Clustering using multiple values of clusters here we didn't do any hyperparameter tuning .

In [ ]: