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).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

[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 [0]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

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

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

from tqdm import tqdm
import os
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?
client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuser
content.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_t
ype=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.t
est%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fw
ww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.go
ogleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [8]:
```
# using SQLite Table to read data.
con = sqlite3.connect('/content/drive/My Drive/Colab Notebooks/databas
e.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 LIMIT 10000""", 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 (10000, 10)
```

Out[8]:

Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenomin

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

In [0]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [10]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[10]:

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	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 [11]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[11]:

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

In [12]:
```
display['COUNT(*)'].sum()
```

Out[12]:
```
393063
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

[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 [13]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[13]:

	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	

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [15]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time"
,"Text"}, keep='first', inplace=False)
final.shape
```

Out[15]: (9564, 10)

In [16]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[16]: 95.64

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 [17]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[17]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenom
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenom
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	

In [0]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [19]:
```
#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()
```

```
(9564, 10)
```

Out[19]:
```
1    7976
0    1588
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.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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 [20]:
```
# 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)
```

```
We have used the Victor fly bait for 3 seasons.  Can't beat it.  Great
product!
==================================================
15 month old loves to eat them on the go! They seem great for a health
y, quick, and easy snack!
==================================================
These chips are truly amazing. They have it all. They're light, crisp,
great tasting, nice texture, AND they're all natural... AND low in fat
and sodium! Need I say more? I recently bought a bag of them at a regul
ar grocery store, and couldn't belive my taste buds. That's why I excit
ed why I saw them here on Amazon, and decided to buy a case!
==================================================
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
These tablets definitely made things sweeter -- like lemons, limes, and
grapefruit.  But it wasn't to the point of sheer amazement.  They also
had an interesting effect on cheeses and vinegar, but still did virtual
ly nothing for beer and wine. The tablets are a bit pricey but they do
work.  If you've got extra money, sure, give them a try, but if you're
looking for some amazing way to get your kids to eat broccoli or someth
ing along those lines then this is not the answer. Fun experiment, but
not life-changing. :)
==================================================
```

In [21]:
```
# 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)
```

```
We have used the Victor fly bait for 3 seasons.  Can't beat it.  Great
product!
```

In [22]:
```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how
-to-remove-all-tags-from-an-element
from bs4 import BeautifulSoup

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

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

soup = BeautifulSoup(sent_1500, 'lxml')
text = soup.get_text()
print(text)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print("="*50)

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

```
We have used the Victor fly bait for 3 seasons.  Can't beat it.  Great
product!
==================================================
15 month old loves to eat them on the go! They seem great for a health
y, quick, and easy snack!
==================================================
These chips are truly amazing. They have it all. They're light, crisp,
great tasting, nice texture, AND they're all natural... AND low in fat
and sodium! Need I say more? I recently bought a bag of them at a regul
ar grocery store, and couldn't belive my taste buds. That's why I excit
ed why I saw them here on Amazon, and decided to buy a case!
==================================================
These tablets definitely made things sweeter -- like lemons, limes, and
grapefruit.  But it wasn't to the point of sheer amazement.  They also
had an interesting effect on cheeses and vinegar, but still did virtual
ly nothing for beer and wine. The tablets are a bit pricey but they do
work.  If you've got extra money, sure, give them a try, but if you're
looking for some amazing way to get your kids to eat broccoli or someth
ing along those lines then this is not the answer. Fun experiment, but
not life-changing. :)
```

In [0]:
```
# 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)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    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 [24]:
```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
These chips are truly amazing. They have it all. They are light, crisp,
great tasting, nice texture, AND they are all natural... AND low in fat
and sodium! Need I say more? I recently bought a bag of them at a regul
ar grocery store, and could not belive my taste buds. That is why I exc
ited why I saw them here on Amazon, and decided to buy a case!
==================================================
```

In [25]:
```
#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)
```

```
We have used the Victor fly bait for  seasons.  Can't beat it.  Great p
roduct!
```

In [26]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
These chips are truly amazing They have it all They are light crisp gre
at tasting nice texture AND they are all natural AND low in fat and sod
ium Need I say more I recently bought a bag of them at a regular grocer
y store and could not belive my taste buds That is why I excited why I
saw them here on Amazon and decided to buy a case
```

In [0]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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 [28]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_reviews = []
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# 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%|██████████| 9564/9564 [00:03<00:00, 2992.36it/s]
```

In [29]:
```
preprocessed_reviews[1500]
```

Out[29]:
```
'chips truly amazing light crisp great tasting nice texture natural low
fat sodium need say recently bought bag regular grocery store could not
belive taste buds excited saw amazon decided buy case'
```

[3.2] Preprocessing Review Summary

In [0]:
```
X=preprocessed_reviews
Y=final['Score']

#No need to split the data in unsupervised learning
```

[4] Featurization

[4.1] BAG OF WORDS

In [31]:
```
#BoW
from sklearn.feature_extraction.text import CountVectorizer
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
vectorizer = CountVectorizer(min_df=10, max_features=None)
vectorizer.fit(X) # fitting on train data ,we cant perform fit on test
 or cv

# we use the fitted CountVectorizer to convert the text to vector
X_bow = vectorizer.transform(X)
print("After vectorizations")
print(X_bow.shape, Y.shape)
```

```
After vectorizations
(9564, 3400) (9564,)
```

[4.2] TF-IDF

In [32]:
```
#below code for converting to tfidf
#i refered sample solution to write this code
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
tf_idf_vect.fit(X)
print("some sample features(unique words in the corpus)",tf_idf_vect.ge
t_feature_names()[0:10])
print('='*50)

X_tf_idf = tf_idf_vect.transform(X)
print("the type of count vectorizer ",type(X_tf_idf))
```

```
some sample features(unique words in the corpus) ['ability', 'able', 'a
ble buy', 'able eat', 'able find', 'able get', 'able order', 'able us
e', 'absolute', 'absolute best']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
```

[4.3] Avg Word2Vec using Word2Vec

In [33]:
```
 # Train your own Word2Vec model using your own text corpus
i=0
list_of_sentance=[]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
for sentance in X:
    list_of_sentance.append(sentance.split())


#training word2vect model
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
# this line of code trains your w2v model on the give list of sentances
w2v_model=Word2Vec(list_of_sentance,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))
print("sample words ", w2v_words[0:50])

#this is the actuall code to convert word2vect to avg w2v:
from tqdm import tqdm
import numpy as np
# average Word2Vec
# compute average word2vec for each review.
sent_vectors = []; # the avg-w2v for each sentence/review is stored in
 this list
for sent in tqdm(list_of_sentance): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u might need to change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors.append(sent_vec)
sent_vectors = np.array(sent_vectors)
print(sent_vectors.shape)
print(sent_vectors[0])
```

```
  2%||          | 154/9564 [00:00<00:06, 1528.76it/s]
```

```
number of words that occured minimum 5 times  5652
sample words  ['used', 'fly', 'bait', 'seasons', 'ca', 'not', 'beat',
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
sample words  ['used', 'fly', 'bait', 'seasons', 'ca', 'not', 'beat',
'great', 'product', 'available', 'traps', 'course', 'total', 'pretty',

'stinky', 'right', 'nearby', 'received', 'shipment', 'could', 'hardly',
'wait', 'try', 'love', 'call', 'instead', 'stickers', 'removed', 'easil
y', 'daughter', 'designed', 'printed', 'use', 'car', 'windows', 'beauti
fully', 'print', 'shop', 'program', 'going', 'lot', 'fun', 'everywher
e', 'like', 'tv', 'computer', 'really', 'good', 'idea', 'final']
```

```
100%|██████████| 9564/9564 [00:08<00:00, 1087.08it/s]
```

```
(9564, 50)
[-0.03222506 -0.18752021  0.25131443  0.34361055 -0.40841735 -0.1600897
1
 -0.14416694  0.0864762  -0.61701137 -0.34762127 -0.10585138 -0.1594020
4
 -0.59857074 -0.15598057 -0.52730368 -0.59269287 -0.3043575   0.4344956
7
  0.06807792 -0.19494663  0.27931633 -0.2872973  -0.19862965  0.1826977
5
  0.02904099 -0.01193258 -0.5064524   0.08303604 -0.01426443 -0.2372252
6
 -0.44969056  0.19913074  0.0494779  -0.11294061  0.271592    0.8750807
2
 -0.14326791 -0.17557652 -0.21552811  0.1298267   0.21347348 -0.0425068
1
 -0.22431983 -0.19339312 -0.11563128  0.03486428 -0.44774428  0.1744941
7
 -0.13472055  0.31688885]
```

[4.4] TFIDF W2V

In [34]:
```
#this is for train data
i=0
list_of_sentance=[]
for sentance in X:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    list_of_sentance.append(sentance.split())


# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
tf_idf_matrix = model.fit_transform(X)
# we are converting a dictionary with word as a key, and the idf as a v
alue
dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))



# 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 ce
ll_val = tfidf

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print(tfidf_sent_vectors.shape)
print(tfidf_sent_vectors[0])
```

```
100%|██████████| 9564/9564 [01:05<00:00, 145.37it/s]
```

```
(9564, 50)
[-0.03222506 -0.18752021  0.25131443  0.34361055 -0.40841735 -0.1600897
1
 -0.14416694  0.0864762  -0.61701137 -0.34762127 -0.10585138 -0.1594020
4
 -0.59857074 -0.15598057 -0.52730368 -0.59269287 -0.3043575   0.4344956
7
  0.06807792 -0.19494663  0.27931633 -0.2872973  -0.19862965  0.1826977
5
  0.02904099 -0.01193258 -0.5064524   0.08303604 -0.01426443 -0.2372252
6
 -0.44969056  0.19913074  0.0494779  -0.11294061  0.271592    0.8750807
2
 -0.14326791 -0.17557652 -0.21552811  0.1298267   0.21347348 -0.0425068
1
 -0.22431983 -0.19339312 -0.11563128  0.03486428 -0.44774428  0.1744941
7
 -0.13472055  0.31688885]
```

[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_)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- 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.
- 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 [0]:
```
from sklearn.cluster import KMeans
import numpy as np
##taking 3 to 10 cluster values in hyper parameter tuning
def Kmeans_Choosing_Best_K(vectorizer_data):
  inertia_value=[]
  k_values=[1,3,5,11,21]
  for i in tqdm(k_values):
    kmeans = KMeans(n_clusters=i, random_state=0).fit(vectorizer_data)
    inertia_value.append(kmeans.inertia_)
  plt.plot(k_values, inertia_value, label='Inertia loss')
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  plt.legend()
  plt.xlabel("K :Cluster")
  plt.ylabel("Inertia")
  plt.title("ERROR PLOTS")
  plt.show()
```

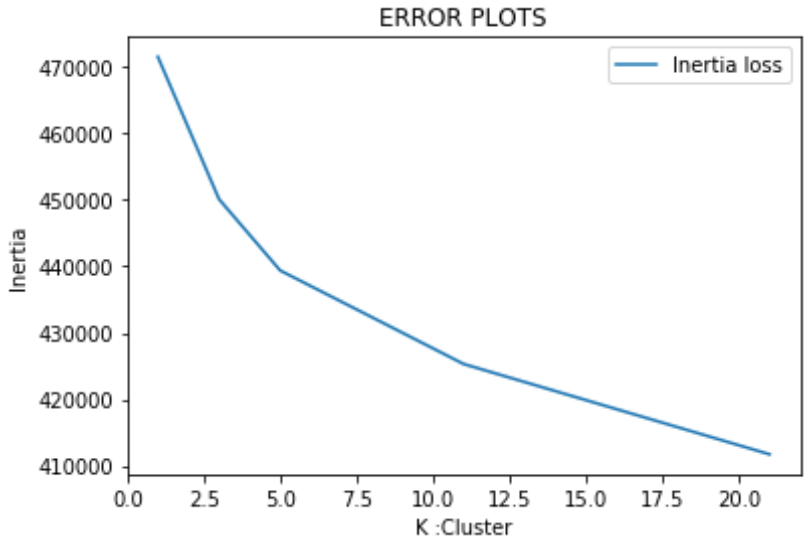
In [0]:
```
def plot_WordCloud(index,data):
  wordcloud = WordCloud(background_color='black',width=1000,height=600)
.generate(data)
  fig = plt.figure(figsize=(10,8))
  plt.imshow(wordcloud)
  plt.axis('off')
  plt.title("World cloud of Cluster {}".format(index))
  plt.tight_layout(pad=0)
  plt.show()
```

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

In [37]:
```
Kmeans_Choosing_Best_K(X_bow)
```

```
100%|██████████| 5/5 [05:41<00:00, 67.75s/it]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Observation :based on the above elbow method graph we can say #cluster = 5 is good

In [0]:
```
#building best model with #cluster=5
kmeans_bow = KMeans(n_clusters=5, random_state=0).fit(X_bow)
```

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

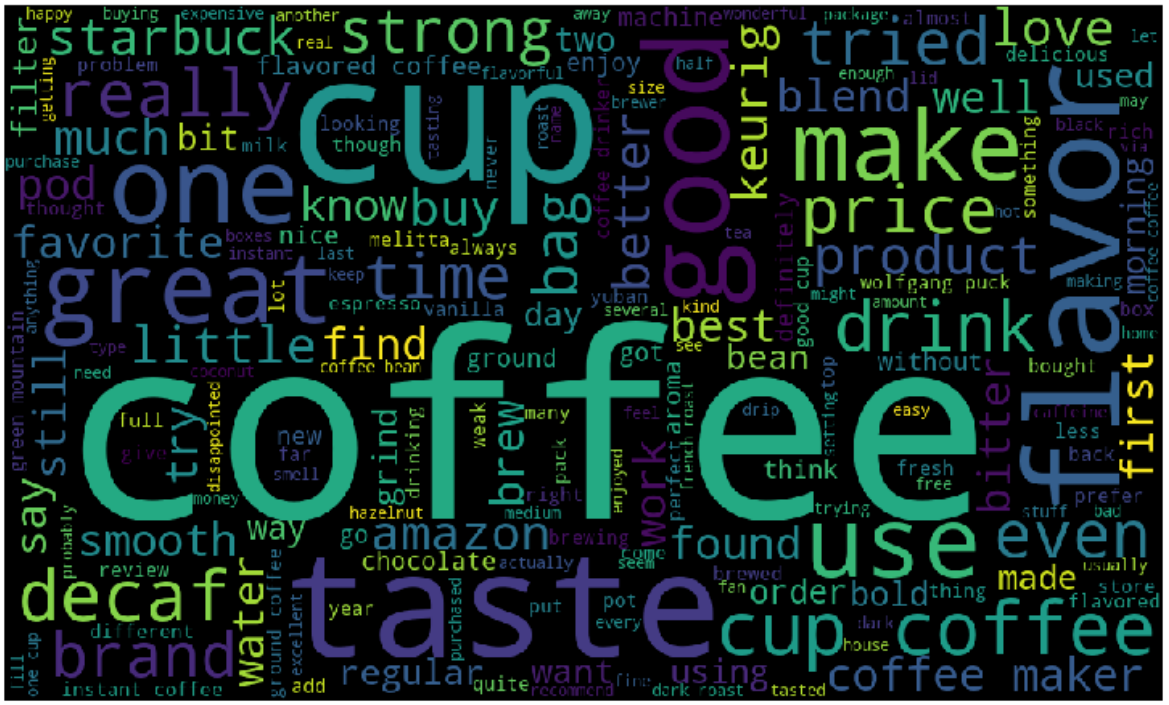
In [39]:
```
#zipping data with cluster labels
data_with_cluster_label = zip(X , kmeans_bow.labels_)

#creating empty lists for each cluster
c0 = []
c1 = []
c2 = []
c3 = []
c4 = []
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#putting reveiws to the corresponding list to the cluster it belongs
for i in range(kmeans_bow.labels_.shape[0]):
  if kmeans_bow.labels_[i] == 0:
    c0.append(X[i])
  elif kmeans_bow.labels_[i] == 1:
    c1.append(X[i])
  elif kmeans_bow.labels_[i] == 2:
    c2.append(X[i])
  elif kmeans_bow.labels_[i] == 3:
    c3.append(X[i])
  elif kmeans_bow.labels_[i] == 4:
    c4.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)
c2_string = ('').join(c2)
c3_string = ('').join(c3)
c4_string = ('').join(c4)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string,c2_string,c3_string,c4_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  plot_WordCloud(index,cluster)
```

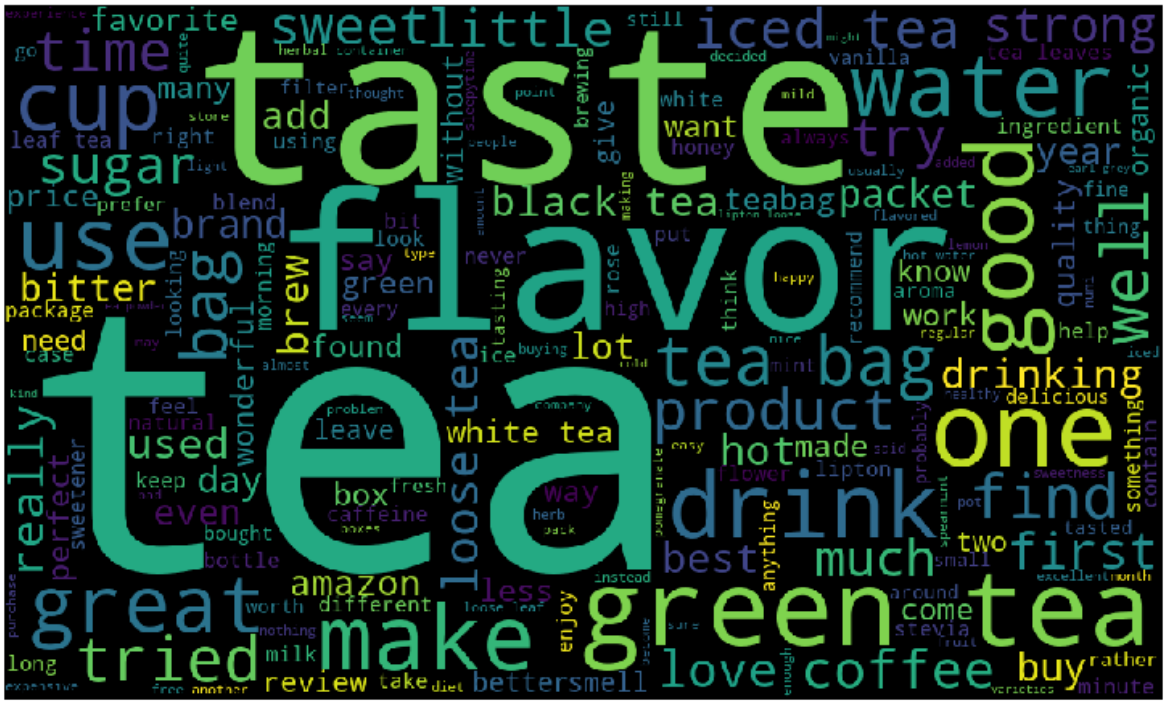
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 0



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 1



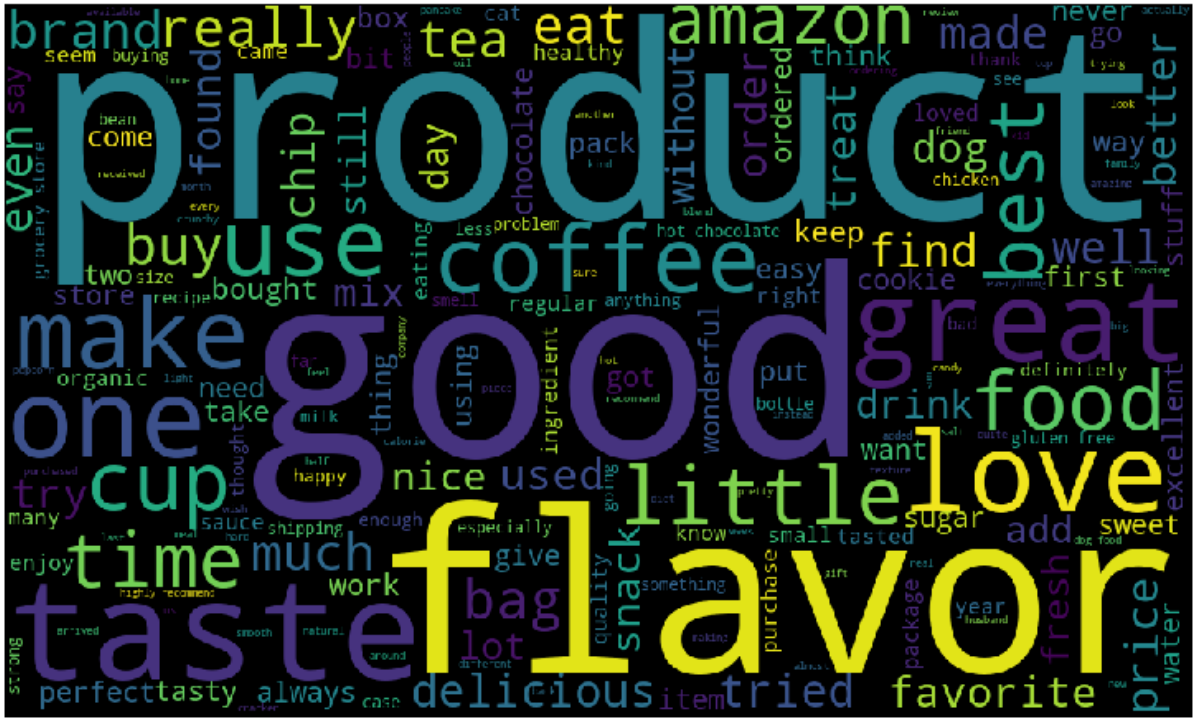
World cloud of Cluster 2

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 3

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 4

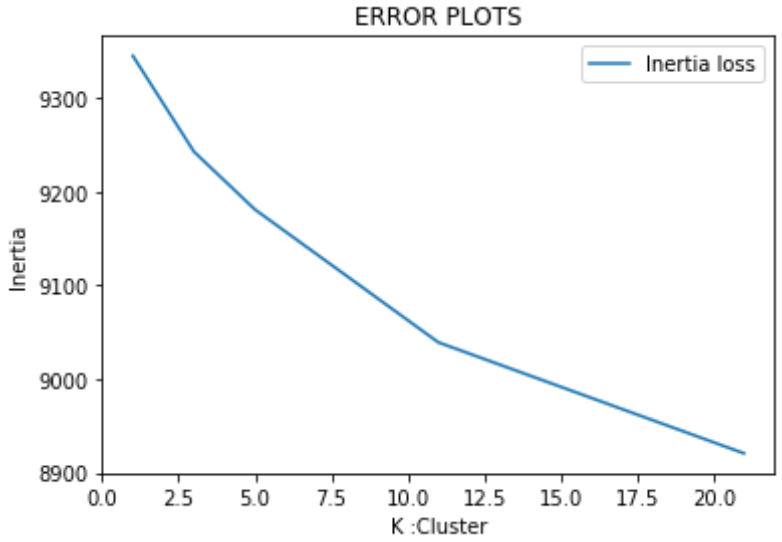


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

In [40]:
```
Kmeans_Choosing_Best_K(X_tf_idf)
```

```
100%|██████████| 5/5 [10:35<00:00, 117.50s/it]
```



Observation :based on the above elbow method graph we can say #cluster = 5 is good

In [0]:
```
#building best model with #cluster=5
kmeans_tfidf = KMeans(n_clusters=5, random_state=0).fit(X_tf_idf)
```

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

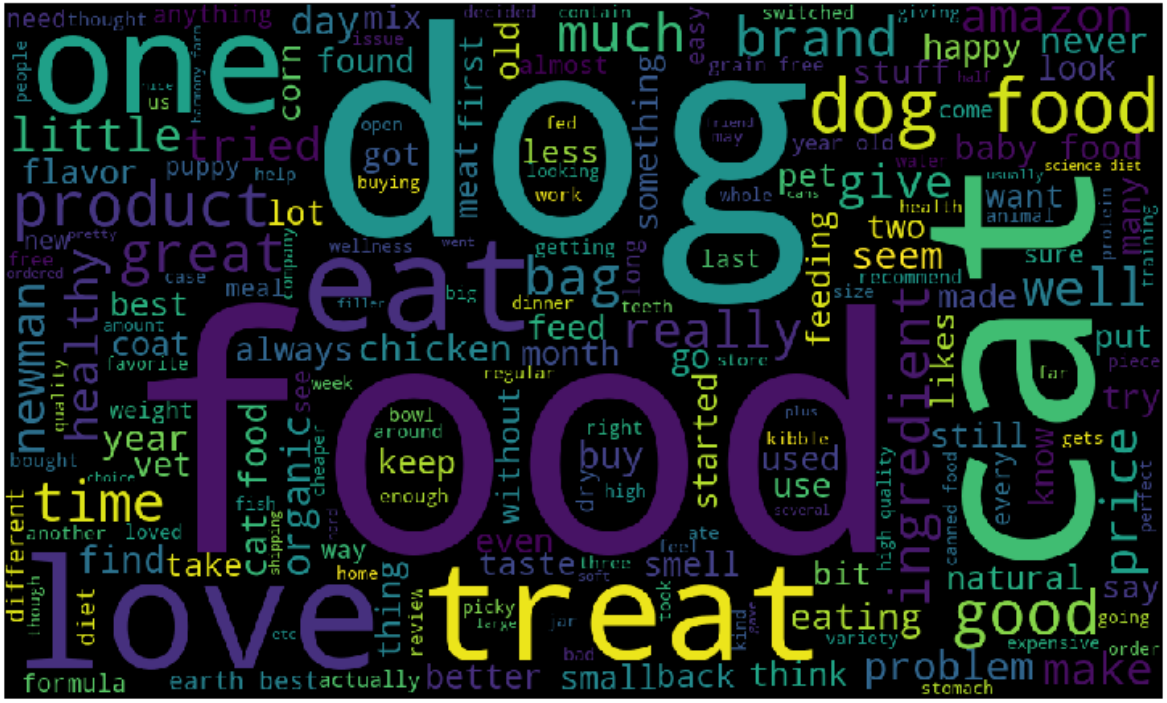
In [43]:
```
#zipping data with cluster labels
data_with_cluster_label = zip(X , kmeans_tfidf.labels_)

#creating empty lists for each cluster
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

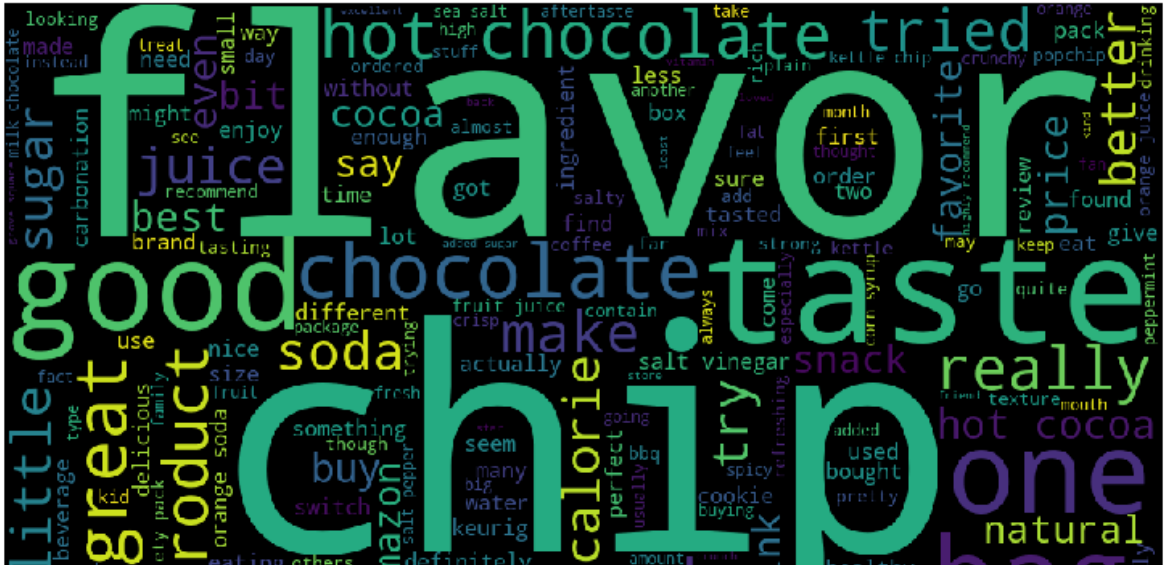
```
c0 = []
c1 = []
c2 = []
c3 = []
c4 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(kmeans_tfidf.labels_.shape[0]):
  if kmeans_tfidf.labels_[i] == 0:
    c0.append(X[i])
  elif kmeans_tfidf.labels_[i] == 1:
    c1.append(X[i])
  elif kmeans_tfidf.labels_[i] == 2:
    c2.append(X[i])
  elif kmeans_tfidf.labels_[i] == 3:
    c3.append(X[i])
  elif kmeans_tfidf.labels_[i] == 4:
    c4.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)
c2_string = ('').join(c2)
c3_string = ('').join(c3)
c4_string = ('').join(c4)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string,c2_string,c3_string,c4_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  plot_WordCloud(index,cluster)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 0

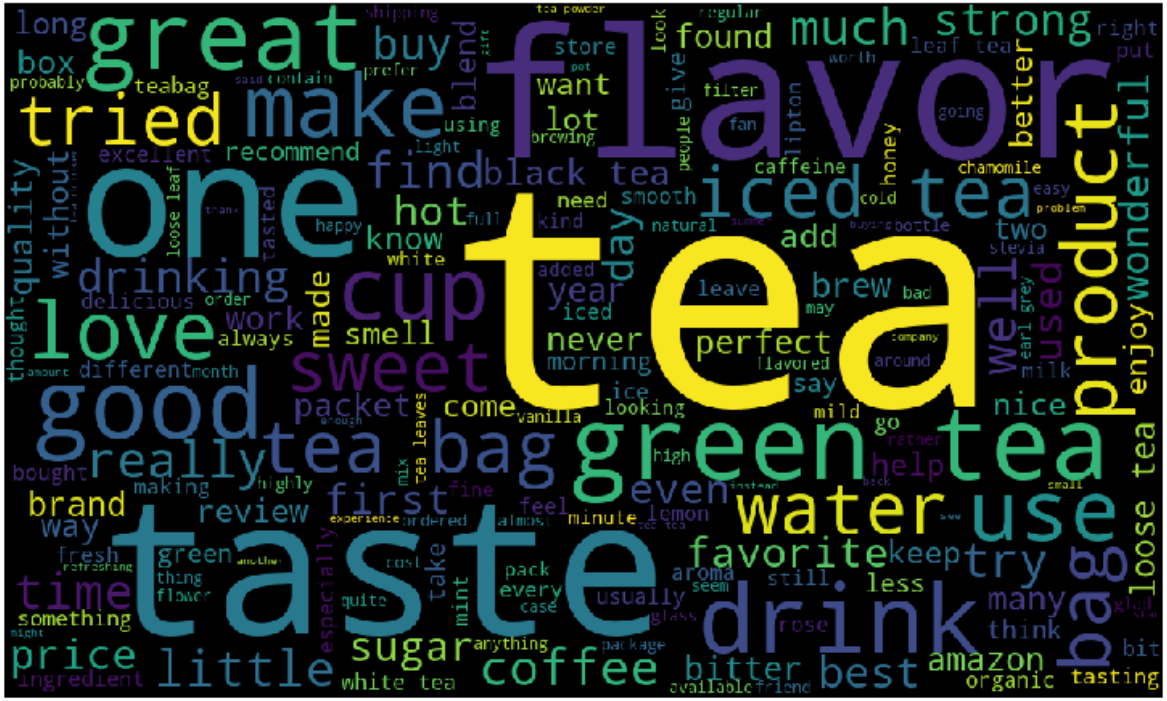


World cloud of Cluster 1



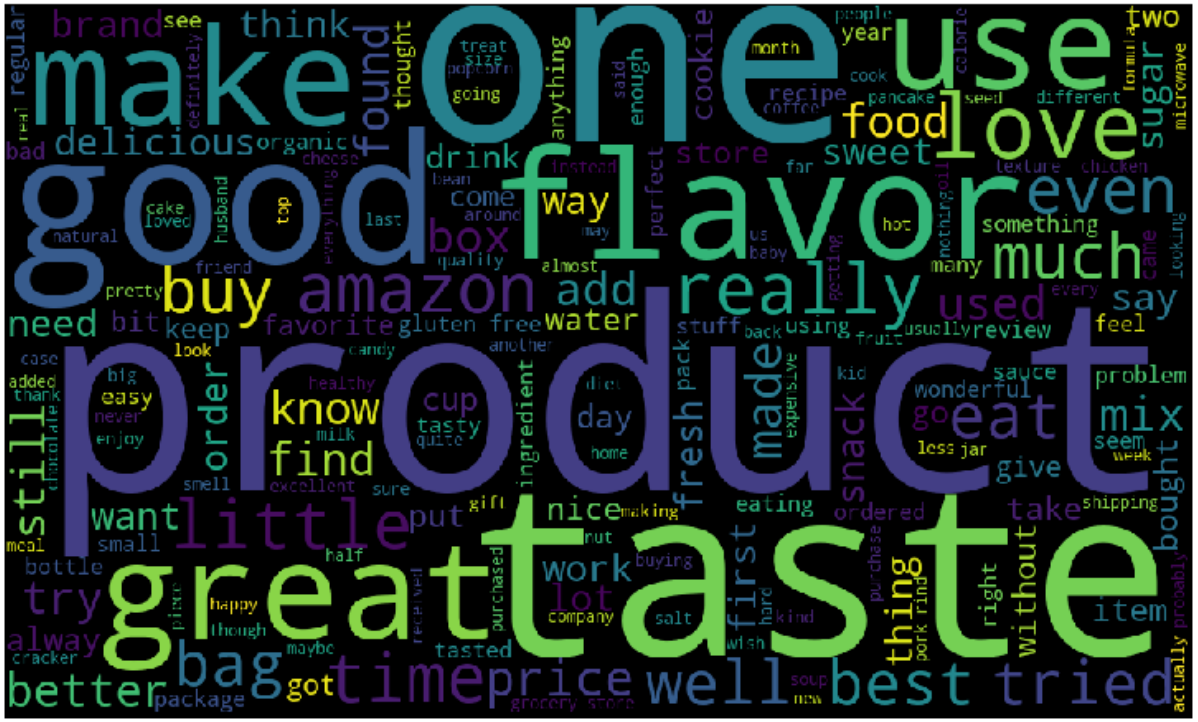
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 2

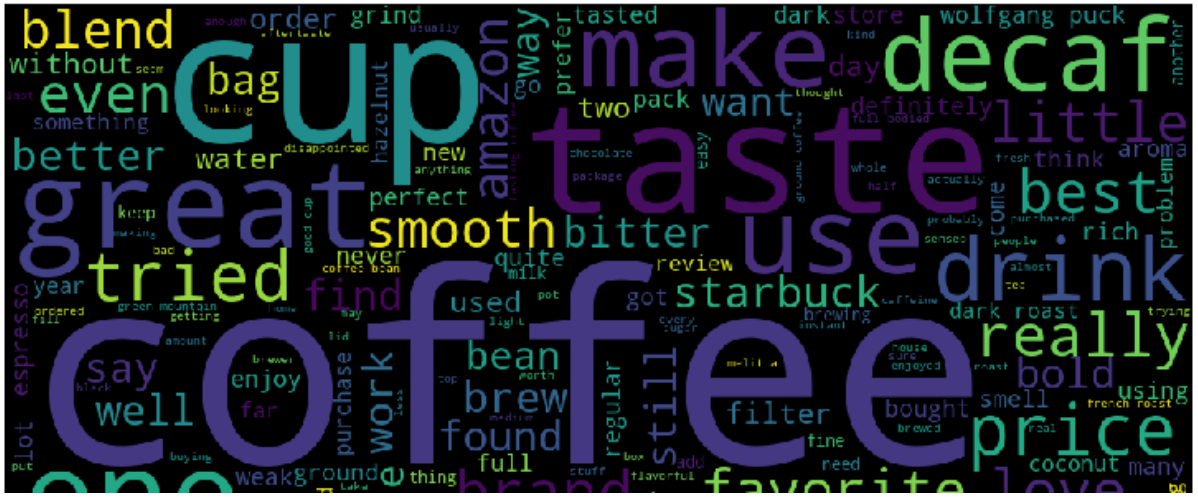


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

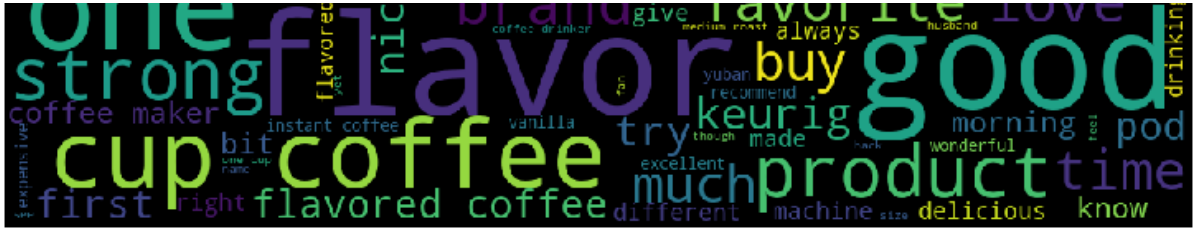
World cloud of Cluster 3



World cloud of Cluster 4



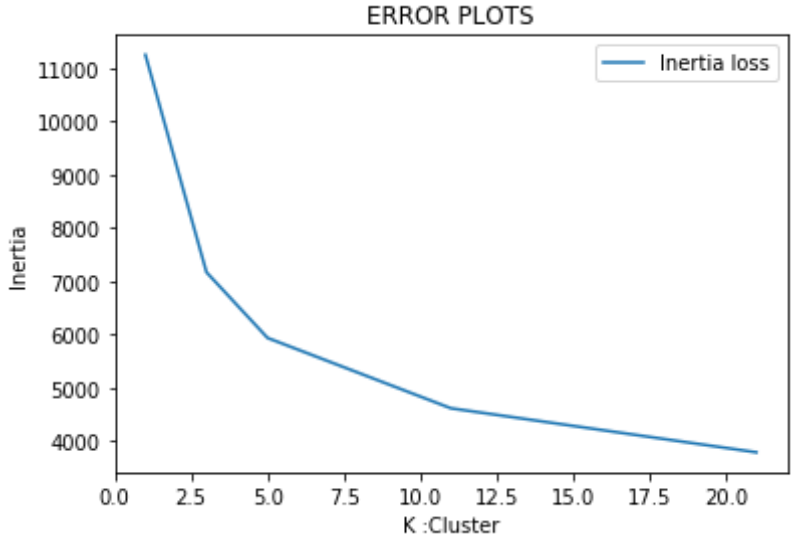
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



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

In [44]:
```
Kmeans_Choosing_Best_K(sent_vectors)
```

```
100%|██████████| 5/5 [00:08<00:00,  2.05s/it]
```



Observation :based on the above elbow method graph we can say #cluster = 5 is good

In [0]:
```
#building best model with #cluster=5
kmeans_avgW2V = KMeans(n_clusters=5, random_state=0).fit(sent_vectors)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

In [47]:
```
#zipping data with cluster labels
data_with_cluster_label = zip(X , kmeans_avgW2V.labels_)

#creating empty lists for each cluster
c0 = []
c1 = []
c2 = []
c3 = []
c4 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(kmeans_avgW2V.labels_.shape[0]):
  if kmeans_avgW2V.labels_[i] == 0:
    c0.append(X[i])
  elif kmeans_avgW2V.labels_[i] == 1:
    c1.append(X[i])
  elif kmeans_avgW2V.labels_[i] == 2:
    c2.append(X[i])
  elif kmeans_avgW2V.labels_[i] == 3:
    c3.append(X[i])
  elif kmeans_avgW2V.labels_[i] == 4:
    c4.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)
c2_string = ('').join(c2)
c3_string = ('').join(c3)
c4_string = ('').join(c4)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string,c2_string,c3_string,c4_string]

#calling the funciton with index value and cluster string
```

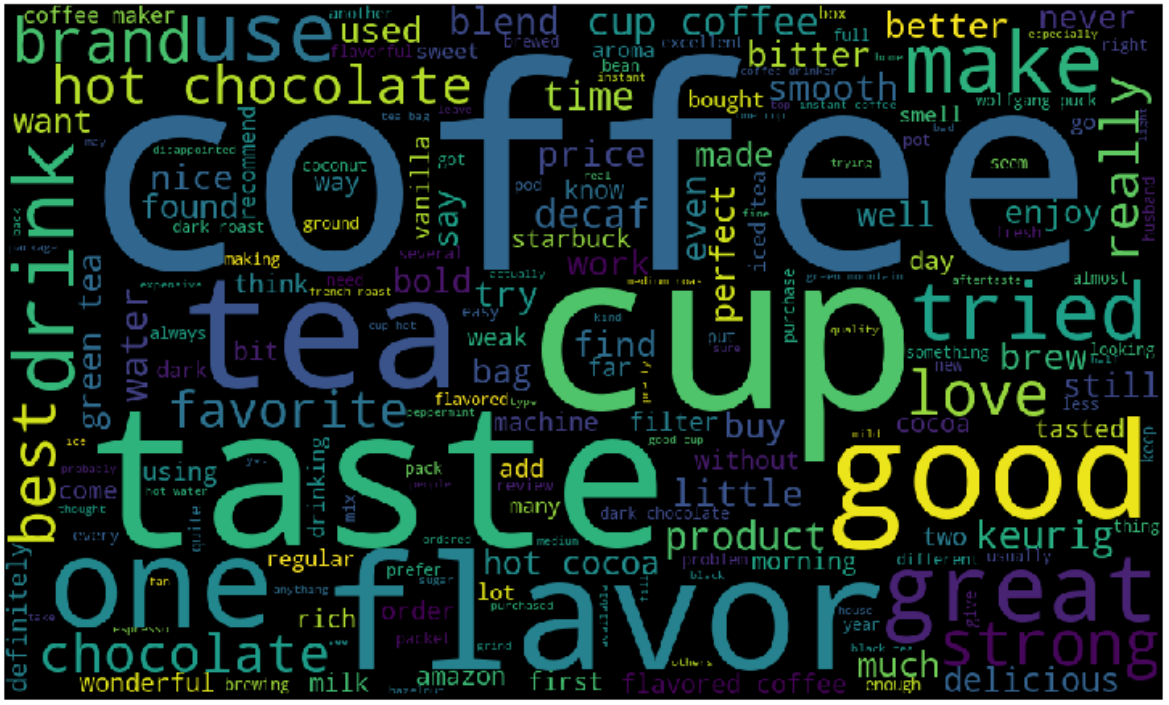
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```python
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  if(len(cluster)>0):
    plot_WordCloud(index,cluster)
```

World cloud of Cluster 0

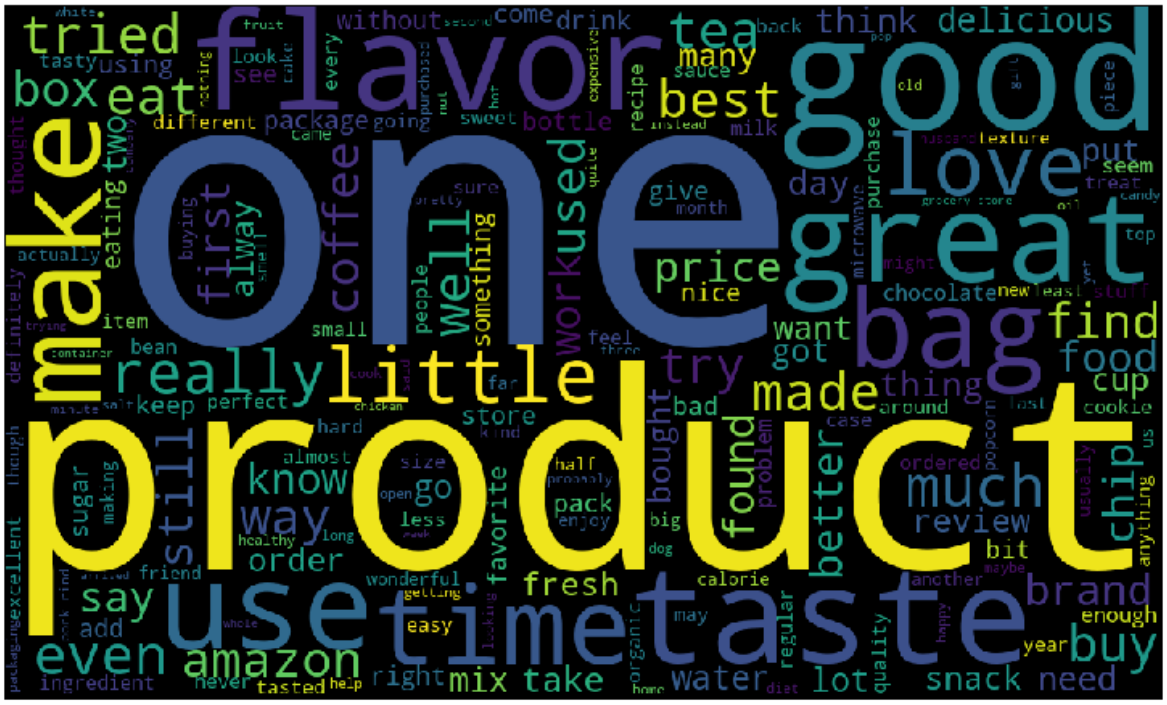
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 1

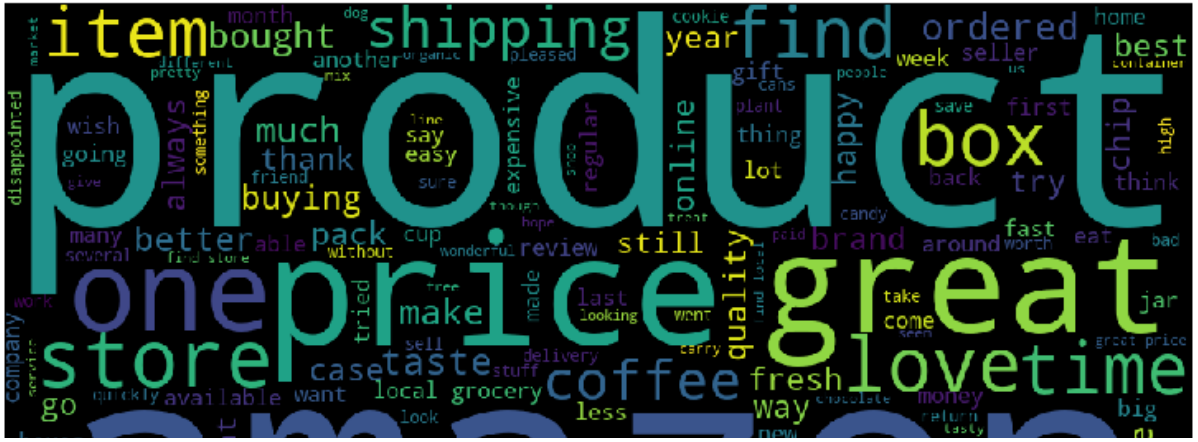


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 2

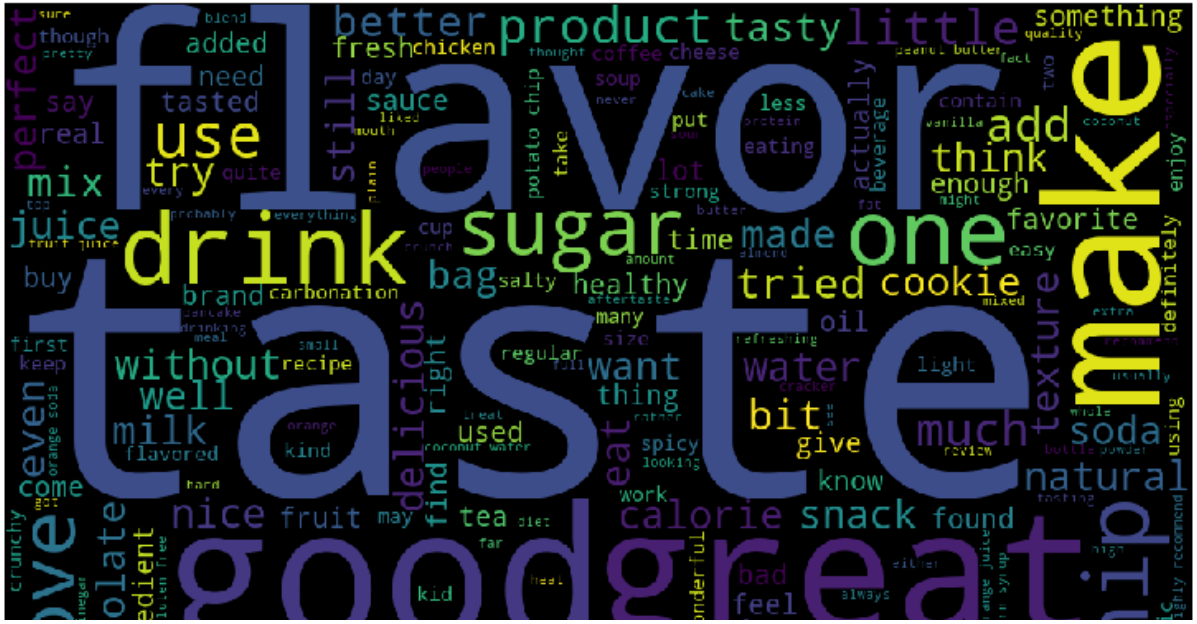


World cloud of Cluster 3



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 4

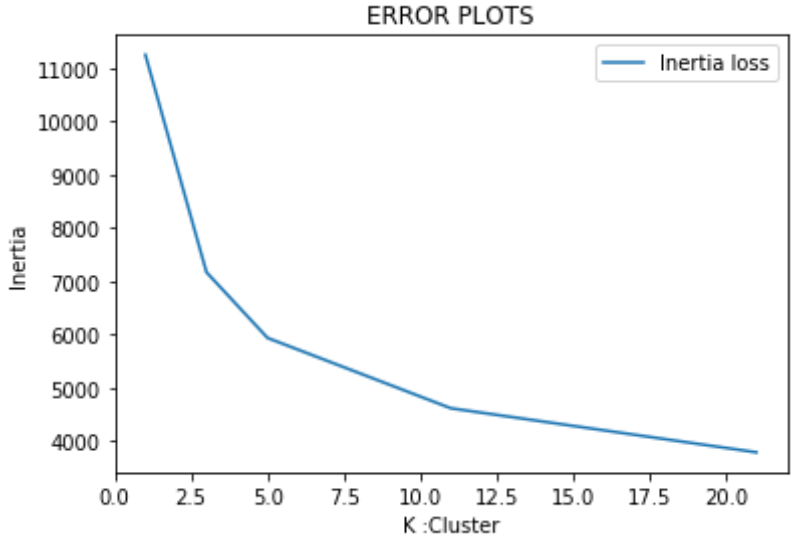


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

In [48]:
```
Kmeans_Choosing_Best_K(tfidf_sent_vectors)
```

```
100%|██████████| 5/5 [00:08<00:00,  2.03s/it]
```



Observation :based on the above elbow method graph we can say #cluster = 5 is good

In [0]:
```
#building best model with #cluster=5
kmeans_tfidf_w2v = KMeans(n_clusters=5, random_state=0).fit(tfidf_sent_
vectors)
```

[5.1.8] Wordclouds of clusters obtained after applying k-means on

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

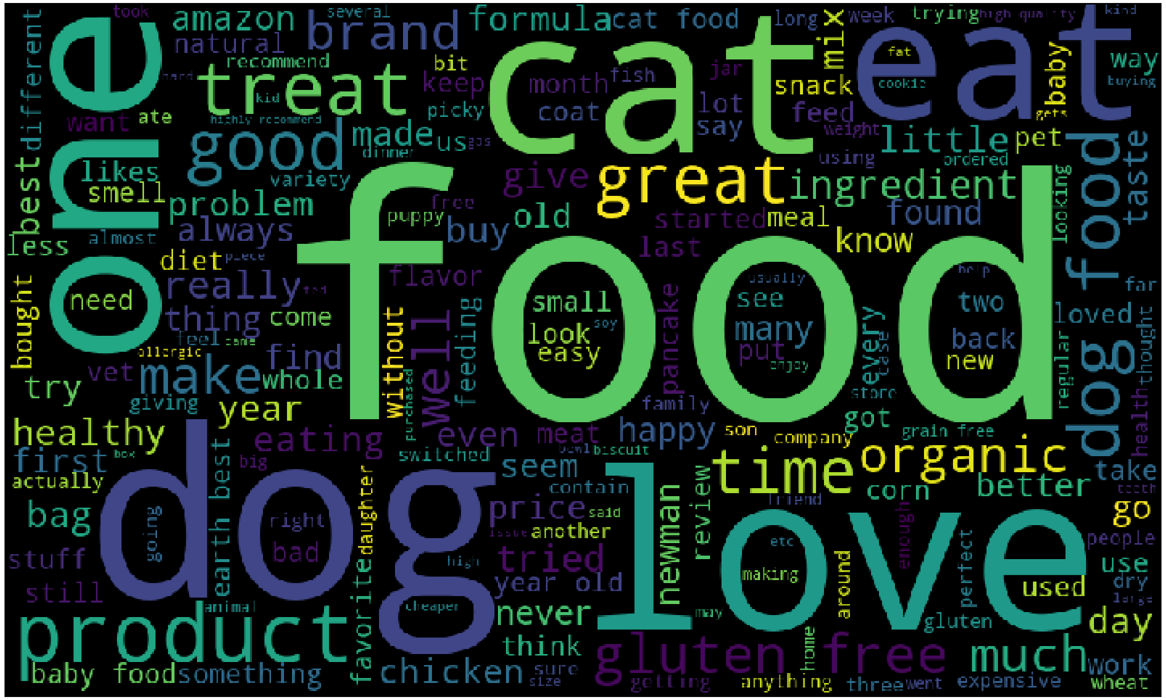
TFIDF W2V SET 4

In [50]:
```
#zipping data with cluster labels
data_with_cluster_label = zip(X , kmeans_tfidf_w2v.labels_)

#creating empty lists for each cluster
c0 = []
c1 = []
c2 = []
c3 = []
c4 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(kmeans_tfidf_w2v.labels_.shape[0]):
  if kmeans_tfidf_w2v.labels_[i] == 0:
    c0.append(X[i])
  elif kmeans_tfidf_w2v.labels_[i] == 1:
    c1.append(X[i])
  elif kmeans_tfidf_w2v.labels_[i] == 2:
    c2.append(X[i])
  elif kmeans_tfidf_w2v.labels_[i] == 3:
    c3.append(X[i])
  elif kmeans_tfidf_w2v.labels_[i] == 4:
    c4.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)
c2_string = ('').join(c2)
c3_string = ('').join(c3)
c4_string = ('').join(c4)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string,c2_string,c3_string,c4_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

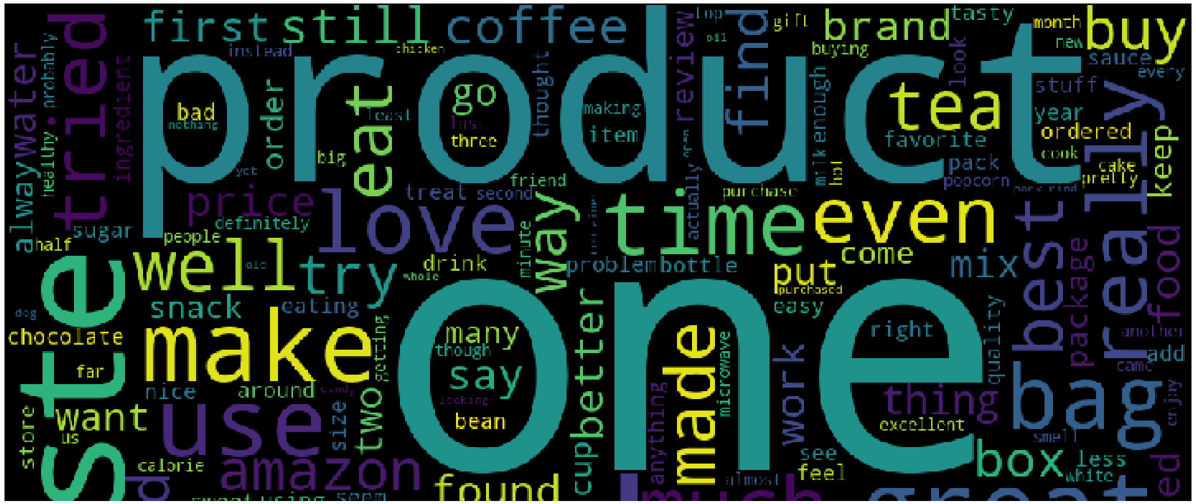
```
    if(len(cluster)>0):
      plot_WordCloud(index,cluster)
```

World cloud of Cluster 0



World cloud of Cluster 1

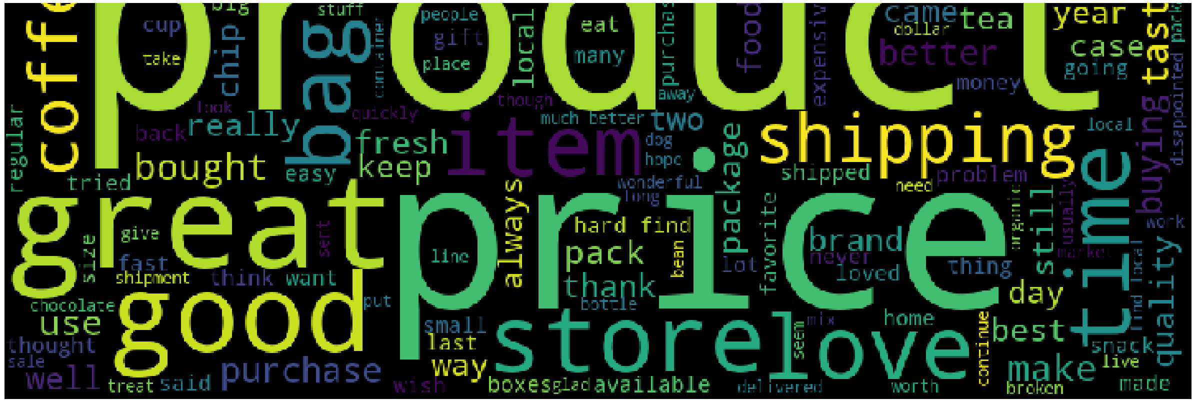
World cloud of Cluster 2



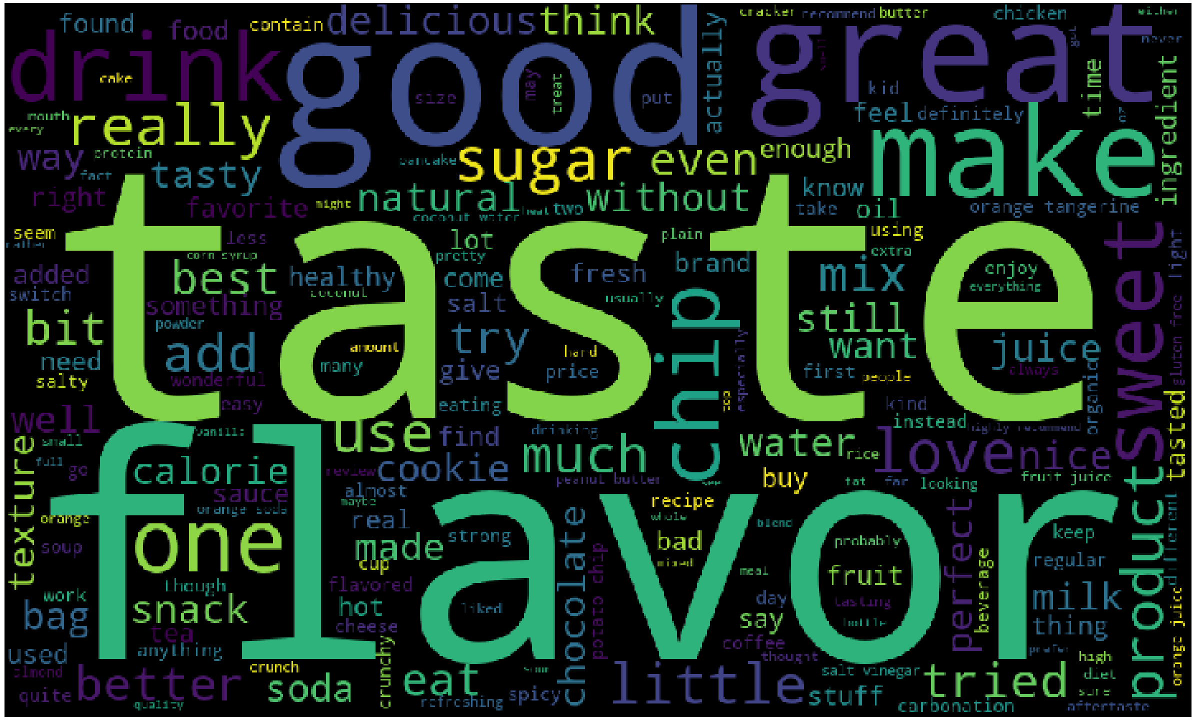
Create PDF in your applications with the Pdfcrowd HTML to PDF API
PDFCROWD

World cloud of Cluster 3

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



World cloud of Cluster 4



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

[5.2] Agglomerative Clustering

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

In [0]:
```
#considering only top 5000 datapoints since agglomerative is very expen
sive
```

In [0]:
```
from sklearn.cluster import AgglomerativeClustering
#First consider number of cluster as 2
agglo_avgw2v_cluster_2 = AgglomerativeClustering(n_clusters=2).fit(sent
_vectors)
```

In [0]:
```
#First consider number of cluster as 5
agglo_avgw2v_cluster_5 = AgglomerativeClustering(n_clusters=5).fit(sent
_vectors)
```

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

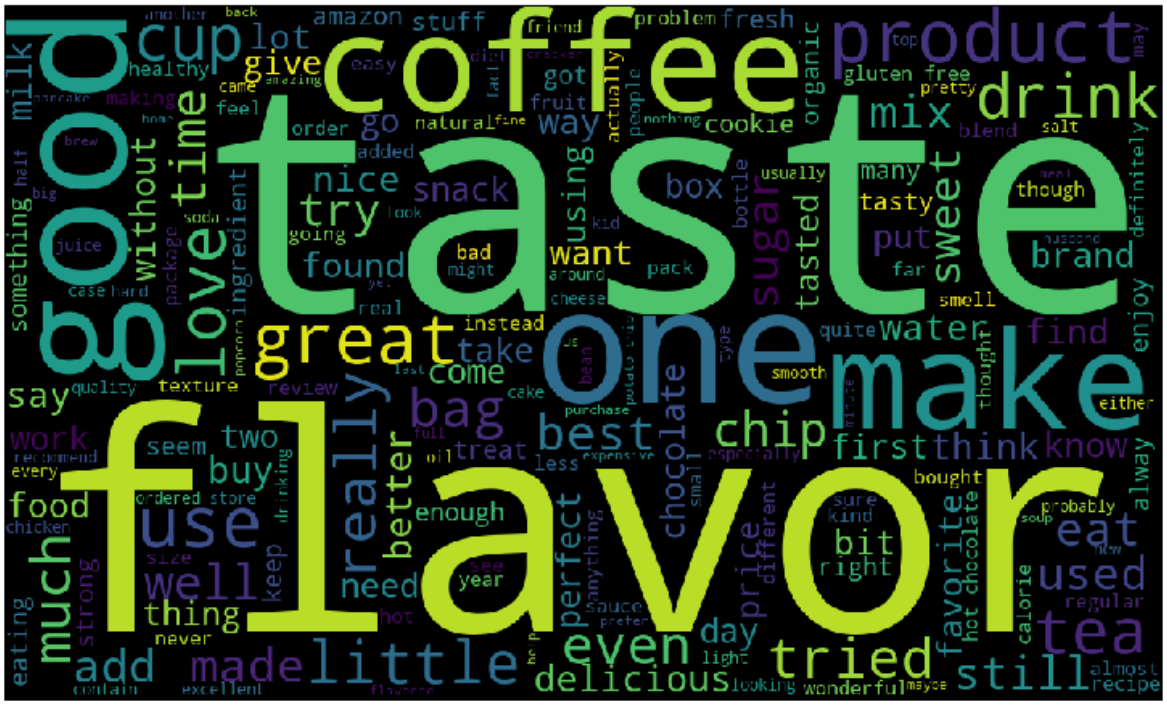
In [55]:
```
#plotting word cloud for #cluster=2
#zipping data with cluster labels
data_with_cluster_label = zip(X , agglo_avgw2v_cluster_2.labels_)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

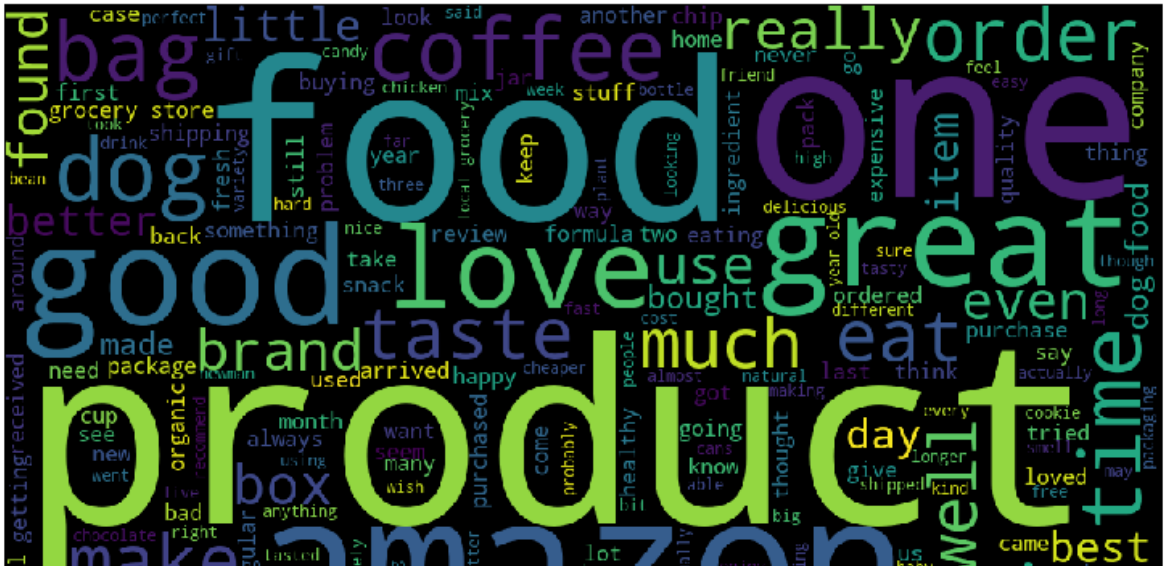
```
#creating empty lists for each cluster
c0 = []
c1 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(agglo_avgw2v_cluster_2.labels_.shape[0]):
  if agglo_avgw2v_cluster_2.labels_[i] == 0:
    c0.append(X[i])
  elif agglo_avgw2v_cluster_2.labels_[i] == 1:
    c1.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  plot_WordCloud(index,cluster)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 0



World cloud of Cluster 1



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Descrpition for cluster 0:This cluster has words which tells about some food product .For example how was the food like good ,delicius etc

Description for cluster 1: This cluster has words like product ,price ,store it has some technical words which are releted to food itself

In [56]:
```
#plotting word cloud for #cluster=5
#zipping data with cluster labels
data_with_cluster_label = zip(X , agglo_avgw2v_cluster_5.labels_)

#creating empty lists for each cluster
c0 = []
c1 = []
c2 = []
c3 = []
c4 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(agglo_avgw2v_cluster_5.labels_.shape[0]):
  if agglo_avgw2v_cluster_5.labels_[i] == 0:
    c0.append(X[i])
  elif agglo_avgw2v_cluster_5.labels_[i] == 1:
    c1.append(X[i])
  elif agglo_avgw2v_cluster_5.labels_[i] == 2:
    c2.append(X[i])
  elif agglo_avgw2v_cluster_5.labels_[i] == 3:
    c3.append(X[i])
  elif agglo_avgw2v_cluster_5.labels_[i] == 4:
    c4.append(X[i])

#converting list values to string
```

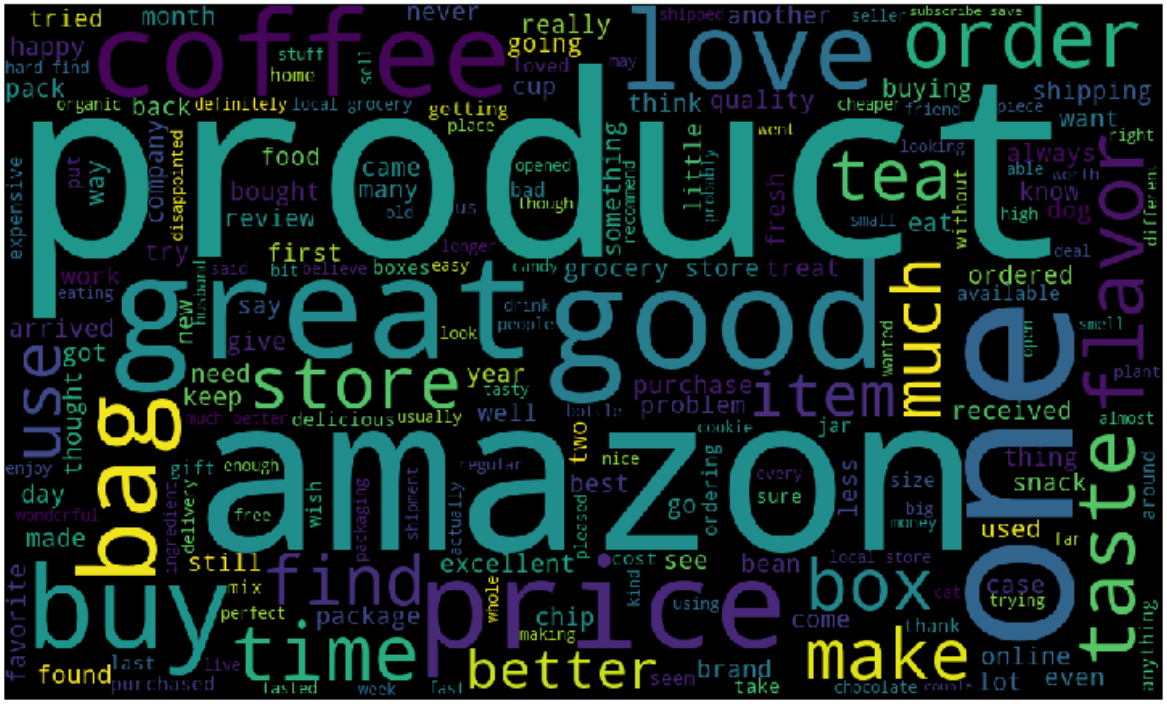
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
c0_string = ('').join(c0)
c1_string = ('').join(c1)
c2_string = ('').join(c2)
c3_string = ('').join(c3)
c4_string = ('').join(c4)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string,c2_string,c3_string,c4_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  if(len(cluster)>0):
    plot_WordCloud(index,cluster)
```

World cloud of Cluster 0

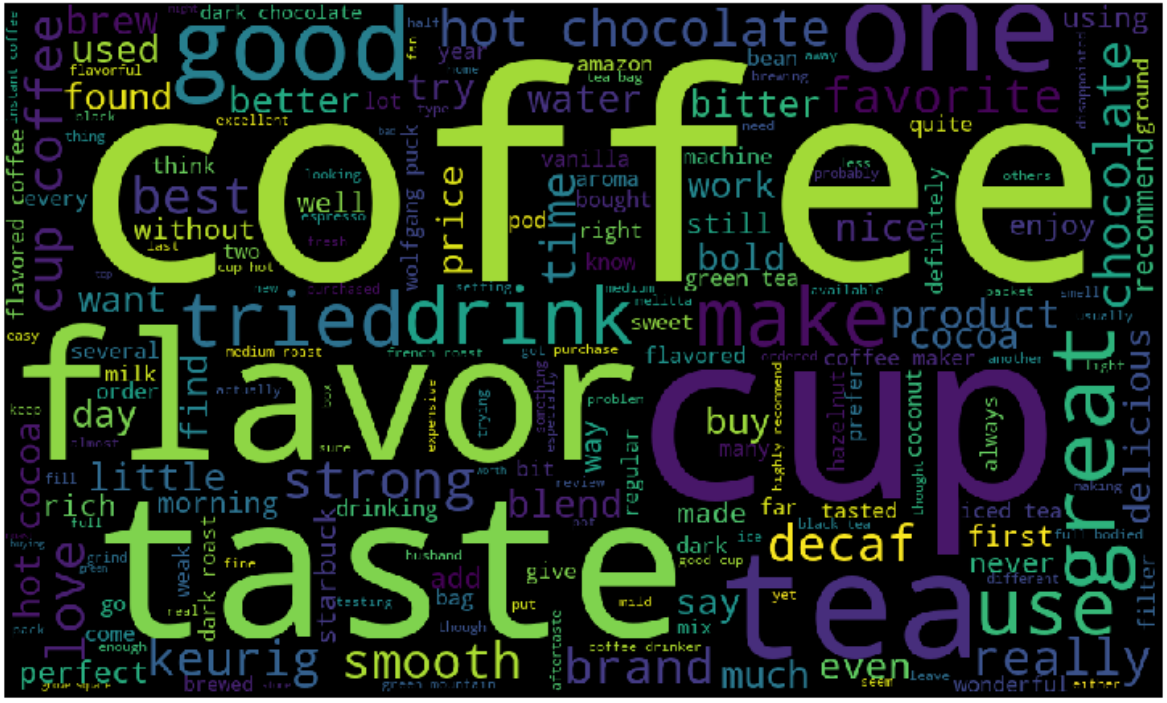
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 1

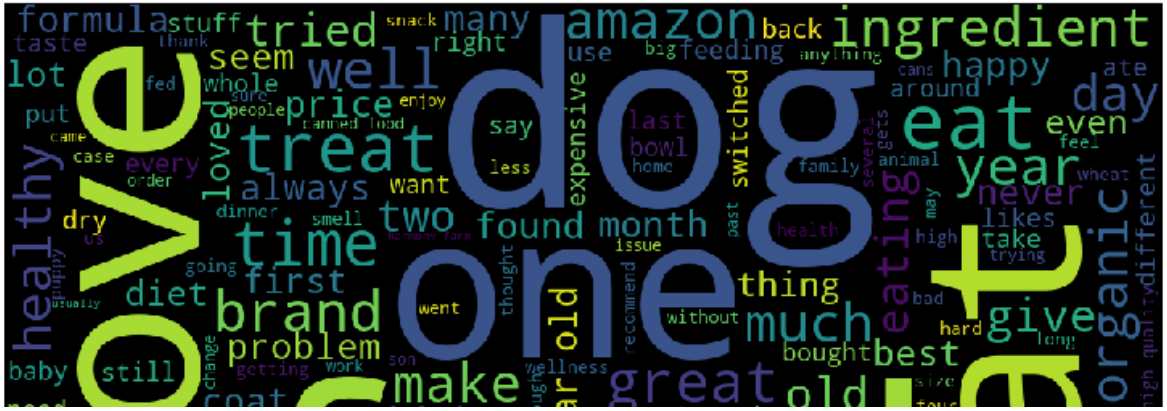


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 2



World cloud of Cluster 3



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 4

Descrpition for cluster 0:This cluster has words which tells about some food product .For example how was the food like good ,delicius etc

Description for cluster 1: This cluster has words like product ,price ,store,order. it has some

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

technical words which are releted to food itself

Description for cluster 2: This cluster words which are fluidy in nature for example water ,coffee ,tea ,cup etc

Description for cluster 3: I observed this cluster has common mix up words ,we cant distinguish to which class and words are releted to which common point

---

Description for cluster 4 : This cluster too has fluidy product words like coconut ,soda ,coffee , tea etc

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

In [0]:
```
#appying agglomerative clustering on tfidf w2v
from sklearn.cluster import AgglomerativeClustering
#First consider number of cluster as 2
agglo_tfidf_w2v_cluster_2 = AgglomerativeClustering(n_clusters=2).fit(t
fidf_sent_vectors)
```

In [0]:
```
#appying agglomerative clustering on tfidf w2v
#First consider number of cluster as 5
agglo_tfidf_w2v_cluster_5 = AgglomerativeClustering(n_clusters=5).fit(t
fidf_sent_vectors)
```

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

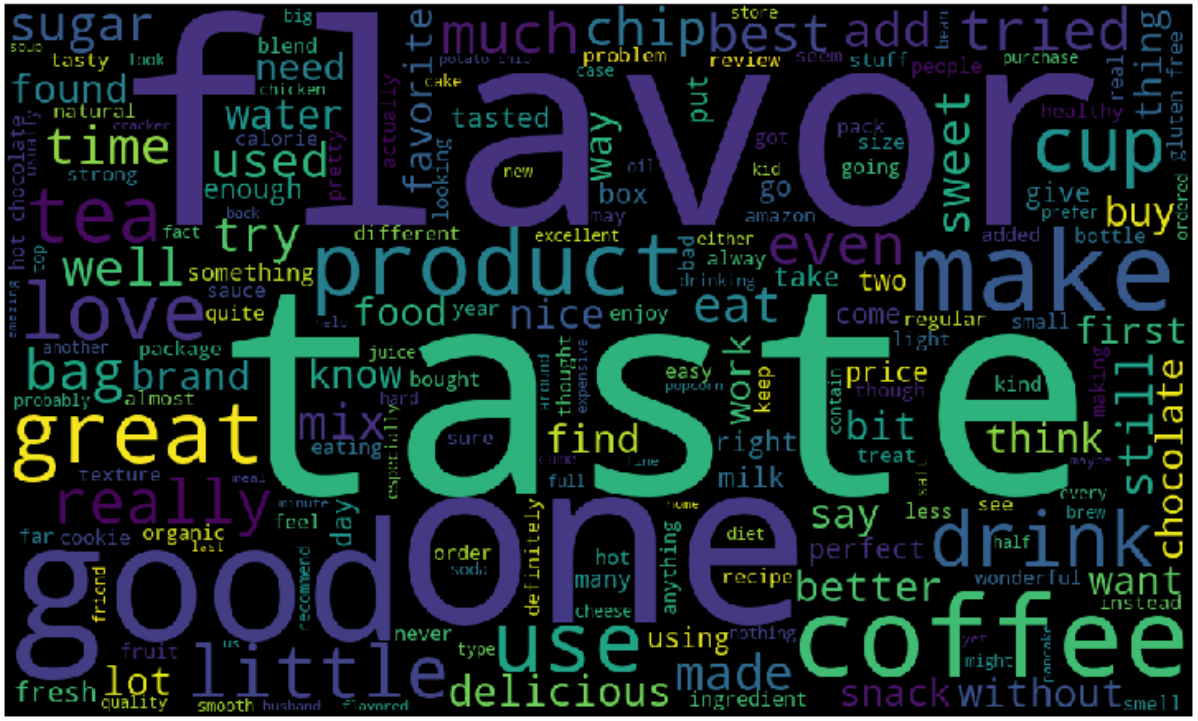
In [61]:
```
#plotting word cloud for #cluster=2
#zipping data with cluster labels
data_with_cluster_label = zip(X , agglo_tfidf_w2v_cluster_2.labels_)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#creating empty lists for each cluster
c0 = []
c1 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(agglo_tfidf_w2v_cluster_2.labels_.shape[0]):
  if agglo_tfidf_w2v_cluster_2.labels_[i] == 0:
    c0.append(X[i])
  elif agglo_tfidf_w2v_cluster_2.labels_[i] == 1:
    c1.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  plot_WordCloud(index,cluster)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 0



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 1

Descrpition for cluster 0:This cluster has words which tells about some food product .For example how was the food like good ,delicius etc

Description for cluster 1: This cluster has words like product ,price ,store,order. it has some technical words which are releted to food itself

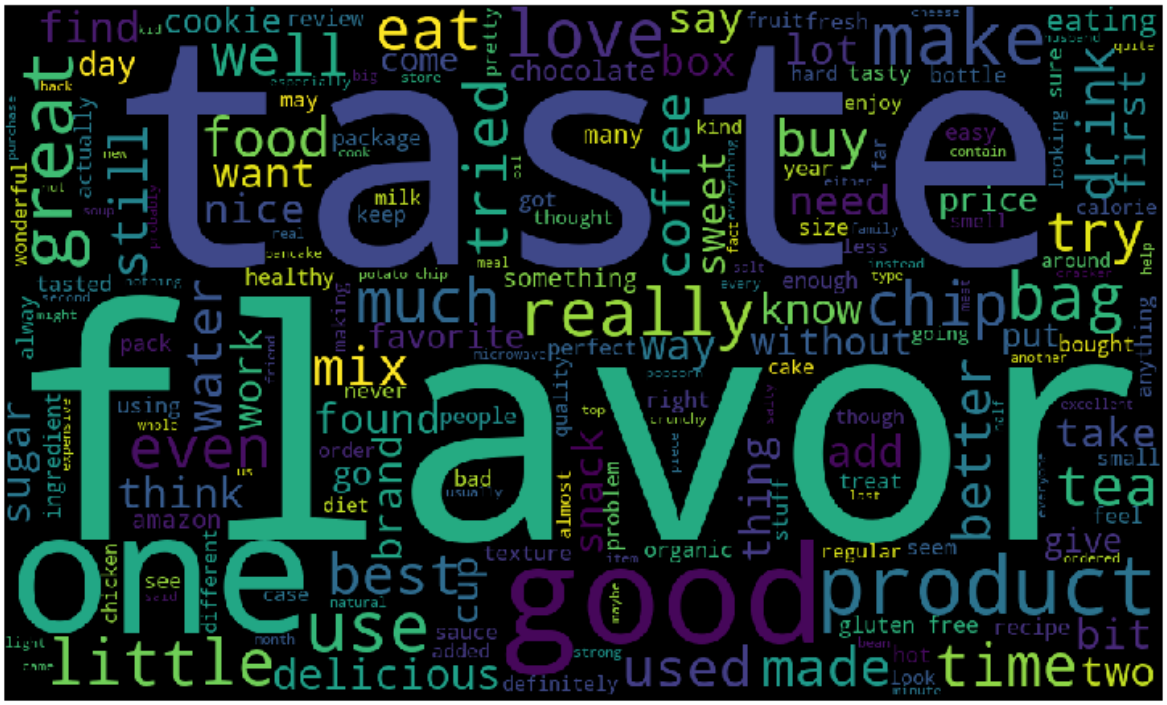
In [62]:
```
#plotting word cloud for #cluster=5
#zipping data with cluster labels
data_with_cluster_label = zip(X , agglo_tfidf_w2v_cluster_5.labels_)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#creating empty lists for each cluster
c0 = []
c1 = []
c2 = []
c3 = []
c4 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(agglo_tfidf_w2v_cluster_5.labels_.shape[0]):
  if agglo_tfidf_w2v_cluster_5.labels_[i] == 0:
    c0.append(X[i])
  elif agglo_tfidf_w2v_cluster_5.labels_[i] == 1:
    c1.append(X[i])
  elif agglo_tfidf_w2v_cluster_5.labels_[i] == 2:
    c2.append(X[i])
  elif agglo_tfidf_w2v_cluster_5.labels_[i] == 3:
    c3.append(X[i])
  elif agglo_tfidf_w2v_cluster_5.labels_[i] == 4:
    c4.append(X[i])

#converting list values to string
c0_string = ('').join(c0)
c1_string = ('').join(c1)
c2_string = ('').join(c2)
c3_string = ('').join(c3)
c4_string = ('').join(c4)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string,c1_string,c2_string,c3_string,c4_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  if(len(cluster)>0):
    plot_WordCloud(index,cluster)
```

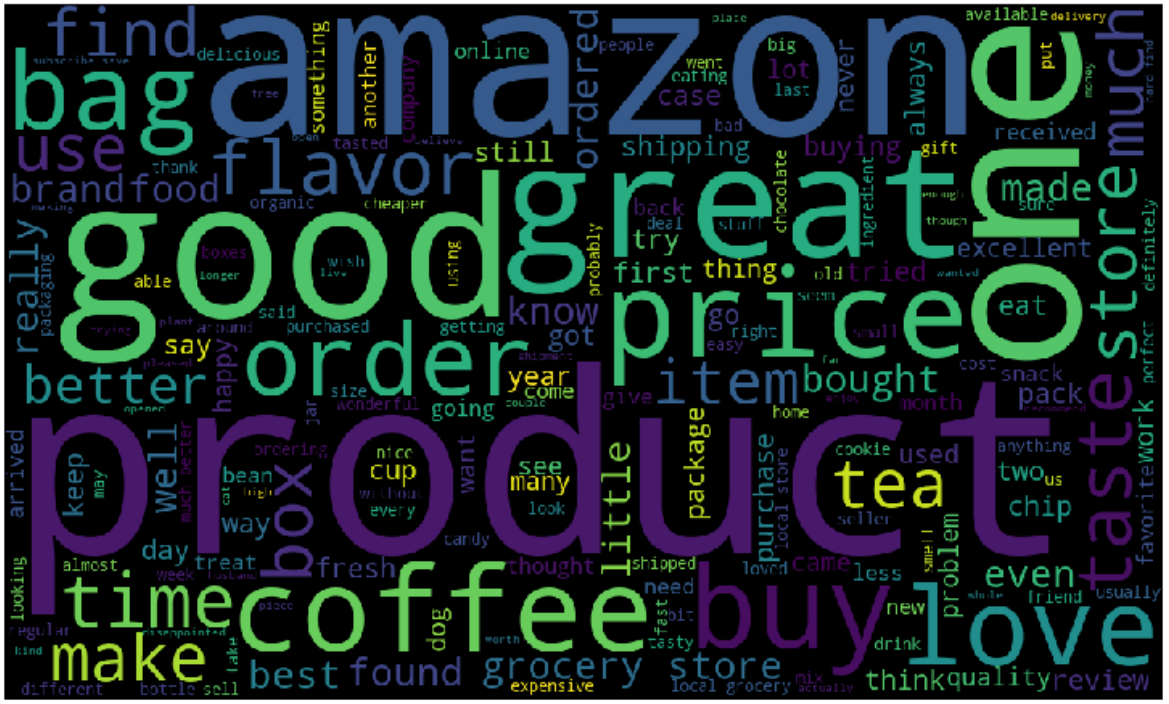
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 0



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

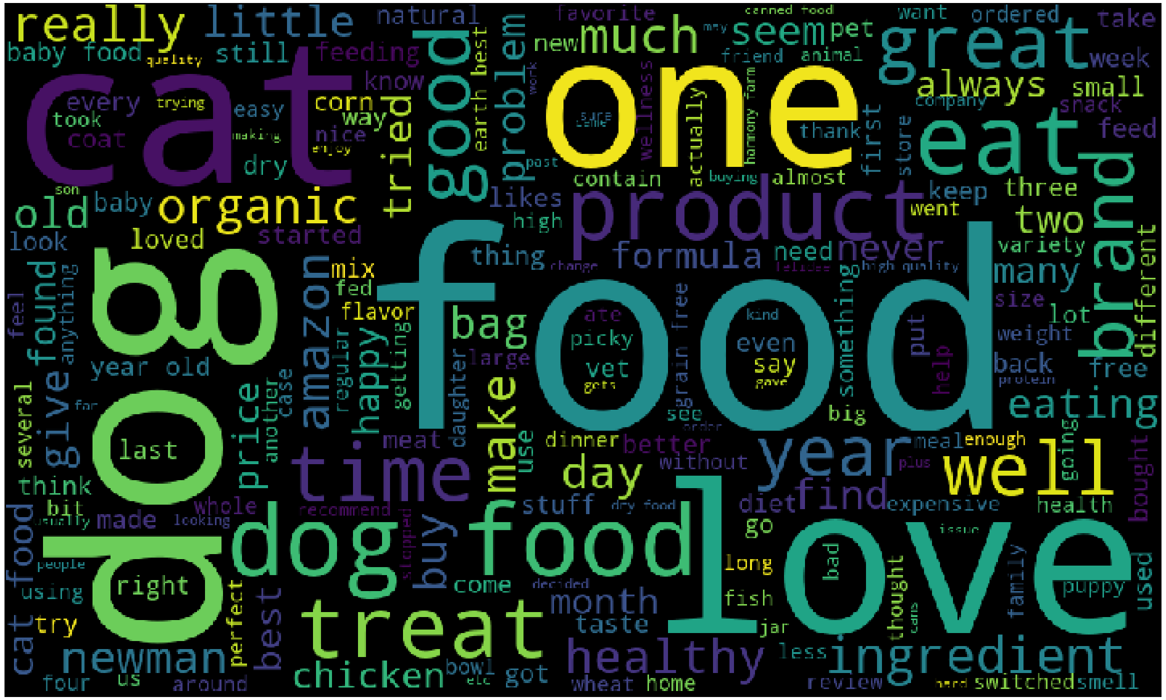
World cloud of Cluster 1



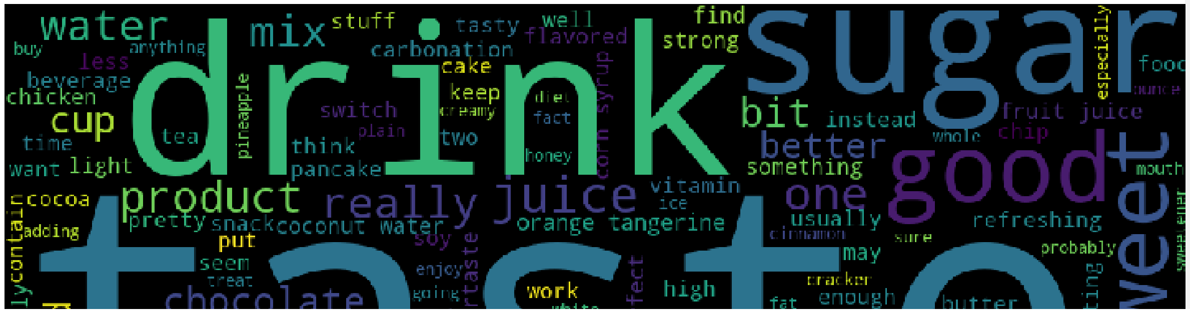
World cloud of Cluster 2

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 3



World cloud of Cluster 4



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Descrpition for cluster 0:This cluster has words which tells about some food product .For example how was the food like good ,delicius etc

Description for cluster 1: This cluster has words like product ,price ,store,order. it has some technical words which are releted to food itself

Description for cluster 2: This cluster words which are fluidy in nature for example water ,coffee ,tea ,cup etc

Description for cluster 3: I observed this cluster has common mix up words ,we cant distinguish to which class and words are releted to which common point

Description for cluster 4 : This cluster too has fluidy product words like coconut ,soda ,coffee , tea etc

[5.3] DBSCAN Clustering

[5.3.1] Applying DBSCAN on AVG W2V, SET 3

In [63]:
```
#The value of minpts is doubtle the size of dimensionality
#we will number of dimensionality
print("dimensionality of data :",sent_vectors.shape[1])
print("Minpts size will be ",(2*(sent_vectors.shape[1])))
```

```
dimensionality of data : 50
Minpts size will be  100
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [64]:
```
#Reference : https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r/4855
#we will apply knn algorithm to get eps value
from sklearn.neighbors import NearestNeighbors
minPts=100
knn_clf = NearestNeighbors(n_neighbors= minPts)
knn_clf.fit(sent_vectors)
```

Out[64]:
```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='minkowski',
                 metric_params=None, n_jobs=None, n_neighbors=100, p=2,
                 radius=1.0)
```

In [0]:
```
dist, indices = knn_clf.kneighbors()
```

In [66]:
```
print(dist.shape)
```

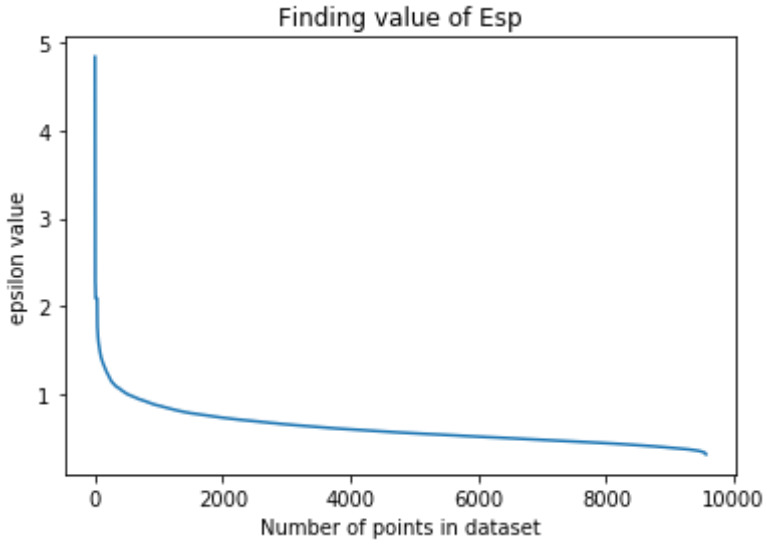
```
(9564, 100)
```

In [0]:
```
dist_Descending_order = sorted(dist[:, minPts-1], reverse = True)
```

In [68]:
```
plt.plot(list(range(1,len(dist_Descending_order)+1)), dist_Descending_order)
plt.xlabel('Number of points in dataset')
plt.ylabel("epsilon value")
plt.title('Finding value of Esp')
```

Out[68]:
```
Text(0.5, 1.0, 'Finding value of Esp')
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



In the above graph we can clearly see the knee point is in the range of 10 to 20

In [0]:
```
#we will consider any point between (10,20)
#lets consider the value of eps =12
```

In [70]:
```
from sklearn.cluster import DBSCAN
dbscan_avgW2V= DBSCAN(eps = 12, min_samples= 100)
dbscan_avgW2V.fit(sent_vectors)
```

Out[70]:
```
DBSCAN(algorithm='auto', eps=12, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=100, n_jobs=None, p=None)
```

In [71]:
```
#Number of points per cluster
print(np.unique(dbscan_avgW2V.labels_))
print("we got only one cluster for it that is :0")
#there is no noisy cluster since no -1 sign in the array
```

```
[0]
we got only one cluster for it that is :0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [72]:
```
#both have same lenght #cross verification
print(len(X))
print(dbscan_avgW2V.labels_.shape)
```

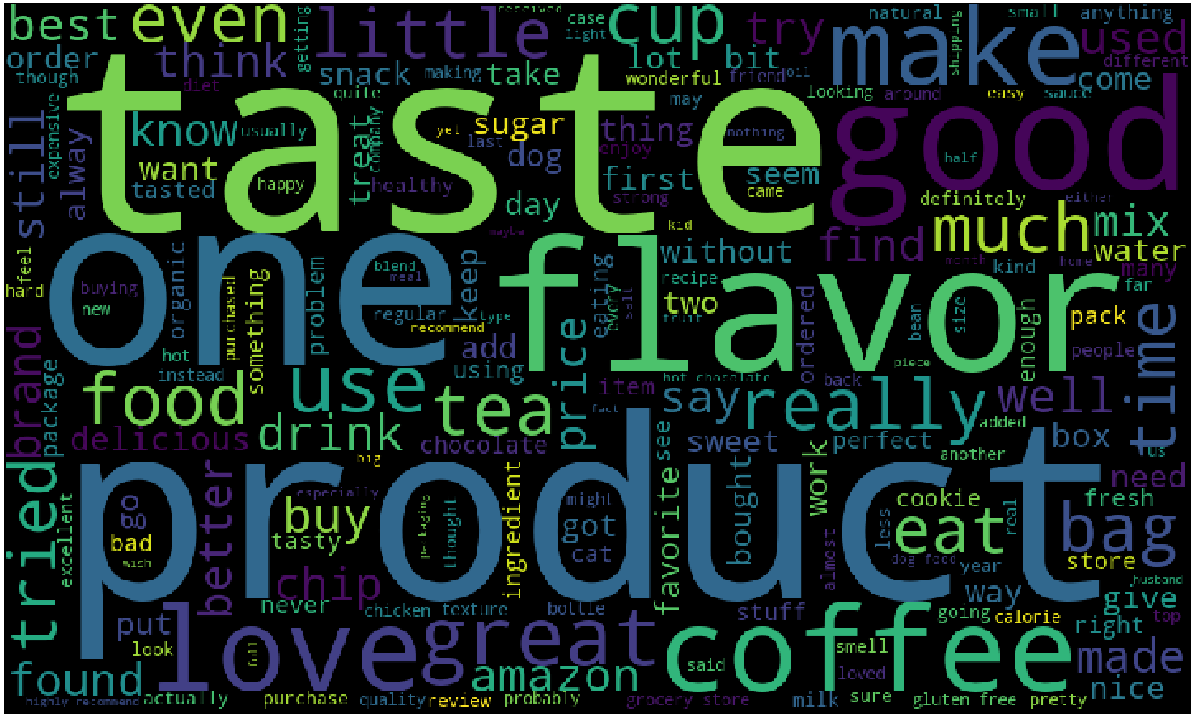
```
9564
(9564,)
```

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

In [73]:
```
#plotting word cloud for #cluster=2
#zipping data with cluster labels
data_with_cluster_label = zip(X , dbscan_avgW2V.labels_)

#creating empty lists for each cluster
c0 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(dbscan_avgW2V.labels_.shape[0]):
  if dbscan_avgW2V.labels_[i] == 0:
    c0.append(X[i])

#converting list values to string
c0_string = ('').join(c0)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  plot_WordCloud(index,cluster)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

World cloud of Cluster 0



Descrpition for cluster 0:This cluster has words which tells about some food product .For example how was the food like good ,delicius etc

[5.3.3] Applying DBSCAN on TFIDF W2V, SET 4

In [74]:
```
from sklearn.cluster import DBSCAN
dbscan_tfidf_W2V= DBSCAN(eps = 12, min_samples= 100)
dbscan_tfidf_W2V.fit(tfidf_sent_vectors)
```

Out[74]:
```
DBSCAN(algorithm='auto', eps=12, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=100, n_jobs=None, p=None)
```

In [75]:
```
#Number of points per cluster
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

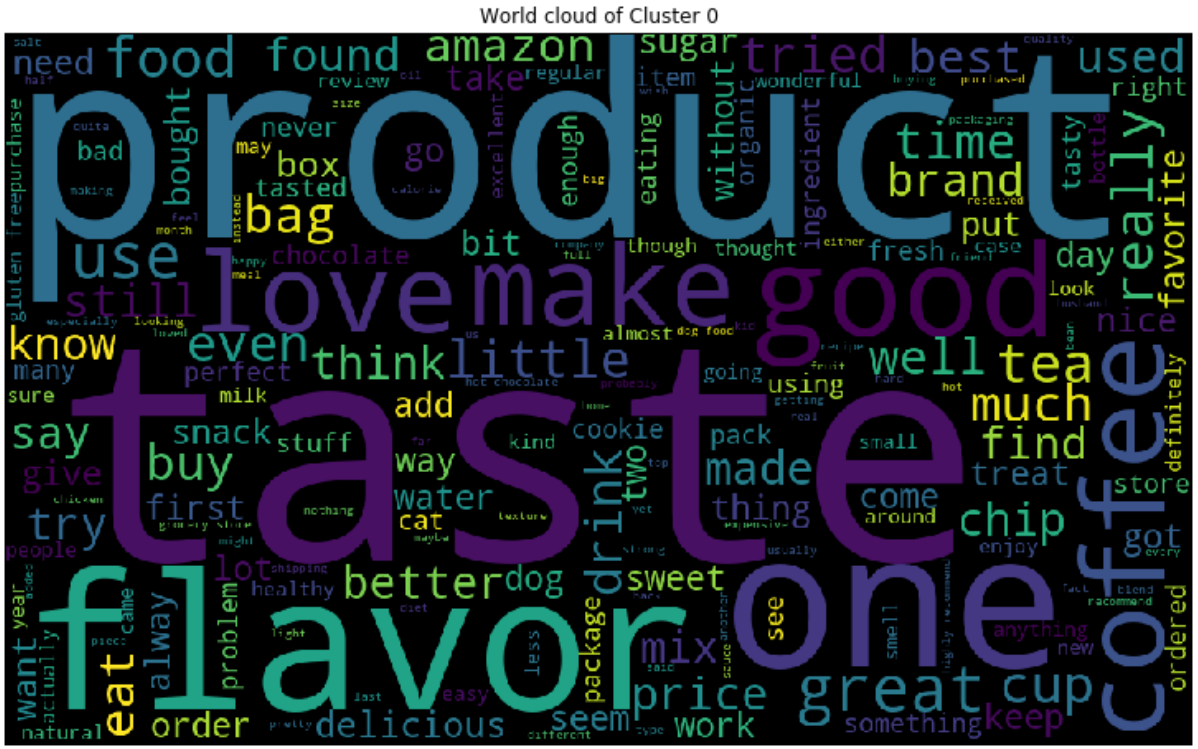
```
print(np.unique(dbscan_tfidf_W2V.labels_))
print("we got only one cluster for it that is :0")
#there is no noisy cluster since no -1 sign in the array
```

```
[0]
we got only one cluster for it that is :0
```

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

In [76]:
```
#plotting word cloud for #cluster=2
#zipping data with cluster labels
data_with_cluster_label = zip(X , dbscan_tfidf_W2V.labels_)

#creating empty lists for each cluster
c0 = []

#putting reveiws to the corresponding list to the cluster it belongs
for i in range(dbscan_tfidf_W2V.labels_.shape[0]):
  if dbscan_tfidf_W2V.labels_[i] == 0:
    c0.append(X[i])

#converting list values to string
c0_string = ('').join(c0)

#making a single list so that it will be helpful to iterate
list_cluster=[c0_string]

#calling the funciton with index value and cluster string
from wordcloud import WordCloud
for index ,cluster in enumerate(list_cluster):
  plot_WordCloud(index,cluster)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Descrpition for cluster 0:This cluster has words which tells about some food product .For example how was the food like good ,delicius etc

[6] Conclusions

In [0]:
```
#Here we cant create pretty table since we dont have any perticular acc
uracy matrx for it
#instead of that ,for each of the word cloud plot we wrote obserercatio
n
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD