In [1]:

```
# importing required libraries
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import nltk
import string

from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
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
from collections import Counter

# ============================== loading libraries ==========================================

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split



from sklearn.feature_extraction.text import CountVectorizer

from prettytable import PrettyTable

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

import itertools
from wordcloud import WordCloud, STOPWORDS

import wordcloud
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
```

```
C:\Users\saurabh.jain\AppData\Local\Continuum\anaconda3\lib\site-packages\smart_open\ssh.py:34: Us
erWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install paramiko`
to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install
paramiko` to suppress')
C:\Users\saurabh.jain\AppData\Local\Continuum\anaconda3\lib\site-packages\gensim\utils.py:1197: Us
erWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [21]:

```
# Loading preprocessed final df
final = pickle.load(open('preprocessed_final', 'rb'))
```

In [22]:

```
X = final['CleanedText'].values
```

Note: To emphasize that this is an unsupervised algorithm, we will leave the labels out of our modeling.

In [4]:

```
type(X)
```

Out[4]:

```
numpy.ndarray
```

Bag Of Words vectorization

In [5]:

```
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer= CountVectorizer(ngram_range=(1,2), min_df=10, max_features=10000)
bow_vectorizer.fit(X) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_bow = bow_vectorizer.transform(X)

# get feature names
feature_names_bow = bow_vectorizer.get_feature_names()

print("After vectorizations")
print(X_bow.shape)
```

```
After vectorizations
(87773, 10000)
```

In [6]:

```
print(type(X_bow))
```

```
<class 'scipy.sparse.csr.csr_matrix'>
```

Standardizing data for BOW

In [7]:

```
# As we are dealing with distance in kmeans and the type of X is a sparse matrix, we will standard
ize the X bag of words data

X_bow = StandardScaler(with_mean=False).fit_transform(X_bow)
```

```
C:\Users\saurabh.jain\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was c
onverted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\saurabh.jain\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was c
onverted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

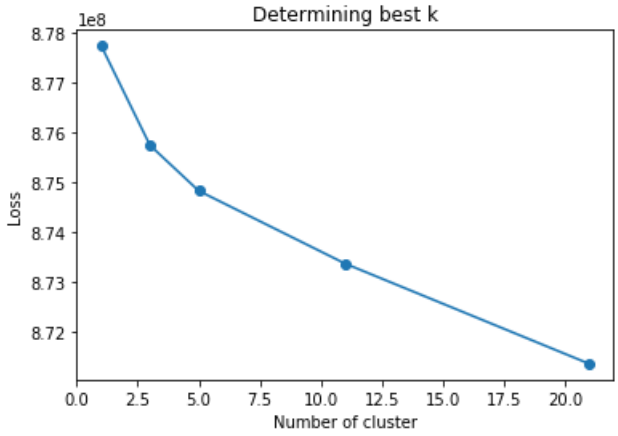
In [9]:

```
def getKmeansBestK(X_data):

    """
```

```
    This function takes in the vectorizer data, iterates over a set of k values and draws a elbow
plot.
    The y axis represents the sum of distances of samples to their closest cluster center.
    """
    k_values = [1, 3, 5, 11, 21]
    sse = {}    # score
    for k in k_values:
        kmeans = KMeans(n_clusters=k).fit(X_data)
        sse[k] = kmeans.inertia_
    plt.figure()
    plt.plot(list(sse.keys()), list(sse.values()), marker='o')
    plt.title("Determining best k")
    plt.xlabel("Number of cluster")
    plt.ylabel("Loss")
    plt.show()
```

In [9]:

```
getKmeansBestK(X_bow)
```



Observation: From the elbow plot, we can see that the best k for bow is **5**

In [8]:

```
# Implementing k-means
kmeans_bow = KMeans(n_clusters= 5, n_jobs=-1, random_state= 507).fit(X_bow)
```

In [72]:

```
filename = 'kmeans_bow_model.sav'
pickle.dump(kmeans_bow, open(filename, 'wb'))
```

In [ ]:

```
# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
```

In [9]:

```
kmeans_bow.labels_
```

Out[9]:

```
array([4, 4, 4, ..., 4, 4, 4])
```

In [10]:

```
result = zip(X , kmeans_bow.labels_)
```

In [11]:

```
result
```

Out[11]:

```
<zip at 0x2673a625fc8>
```

In [12]:

```
X
```

Out[12]:

```
array(['bought apartment infested fruit flies hours trap attracted many flies within days
practically gone may not long term solution flies driving crazy consider buying one caution surfac
e sticky try avoid touching',
       'really good idea final product outstanding use decals car window everybody asks bought
decals made two thumbs',
       'received shipment could hardly wait try product love slickers call instead stickers
removed easily daughter designed signs printed reverse use car windows printed beautifully print s
hop program going lot fun product windows everywhere surfaces like tv screens computer monitors',
       ...,
       'tried orange iced coffee morning really liked going place order today',
       'excellent smooth taste one loves chocolate enjoy drink almost good international house dark
mayan chocolate coffee',
       'purchased product local store ny kids love quick easy meal put toaster oven toast min ready
eat strongly recommend'],
      dtype=object)
```

In [13]:

```
result_df = pd.DataFrame(result)
```

In [14]:

```
result_df.head()
```

Out[14]:

	0	1
0	bought apartment infested fruit flies hours tr...	4
1	really good idea final product outstanding use...	4
2	received shipment could hardly wait try produc...	4
3	nothing product bother link top page buy used ...	4
4	love stuff sugar free not rot gums tastes good...	4

In [15]:

```
result_df.rename(columns={0: "CleanedText", 1: "labels"}, inplace= True)
```

In [16]:

```
result_df.head()
```

Out[16]:

	CleanedText	labels
0	bought apartment infested fruit flies hours tr...	4
1	really good idea final product outstanding use...	4
2	received shipment could hardly wait try produc...	4
3	nothing product bother link top page buy used ...	4
4	love stuff sugar free not rot gums tastes good...	4

In [22]:

```
# Number of reviews for each label
result_df['labels'].value_counts()
```

Out[22]:

```
4    75685
3    11715
0      360
1        8
2        5
Name: labels, dtype: int64
```

In [34]:

```
# Based on the values of label column, we will insert the reviews to a separate list respectively.
# We then pass this list as str to wordcloud

reviews = result_df['CleanedText'].values

# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []

for i in range(kmeans_bow.labels_.shape[0]):
    if kmeans_bow.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif kmeans_bow.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif kmeans_bow.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif kmeans_bow.labels_[i] == 3:
        cluster3.append(reviews[i])

    else :
        cluster4.append(reviews[i])
```

In [68]:

```
len(cluster2)
```

Out[68]:

```
5
```

In [70]:

```
cluster2_str
```

Out[70]:

```
'fuzzy wuzzy summary recommended warm fuzzies received single bar morning good thing already ate b
reakfast nibbled instead chomped morning coffee always liked newman line foods taste good made
high quality ingredients foundation donates hundreds millions dollars charities throughout world l
ove movies paul newman walked philanthropic walk helping make world better place cocoa dark
chocolate bar organic ingredient list free artificial sweetener extraneous additives fillers
common chocolates candies foods days note time review writing ingredients nutrition facts listed a
mazon site dark chocolate bar actually copied another newman chocolate bar actual wrapper ingredie
nts listing organic dark chocolate organic evaporated cane juice organic chocolate liquor organic
cocoa butter organic soy lecithin emulsifier organic vanilla wrapper nutrition facts listing
amazon corresponding nutrition facts listing actually apply newman milk chocolate bar parentheses
calories fat fat fat trans fat carbohydrate c newman organics milk chocolate bar vitamins calcium
iron due use milk dark chocolate fiber less sodium many people not consider chocolate high fiber f
ood dark chocolate not milk chocolate ranks among high fiber foods dark chocolate loaded
```

flavonoids act antioxidants keeping heart healthy reducing oxidative stress flavonoids also balance hormones body activate endothelial nitric oxide synthase levels body biochemical gibberish translates helps regulate blood pressure studies also shown dark chocolate perhaps due increased fiber content makes feel filled giving feeling satiety compared eating milk chocolate time review writing amazon not yet priced chocolate even priced higher dark chocolates offered larger candy factories would still buy believe newman cause charities also love dark chocolate combined nuts berries chocolove dark chocolates hope newman dark chocolates line also eventually includes love dark chocolates cocoa content goes beyond not sweet subtle bitterness compared dark chocolates lower cocoa content cocoa content newman bar strikes happy balance sweetness appeal accustomed milk chocolate sweetness chocolate not sweet cocoa dark chocolates many dark chocolates add higher amounts extra sugar bars flavor chocolate nice slightly fibrous texture not expect sweet sweetened dark chocolates market else may think tastes bit bland compared likewise people think heavily salted food tastes better lightly salted food evaporated cane juice used sweetener chocolate healthier alternative processed sugar included many chocolate bars including belgium made chocolove bars love refined sugar cane juice come sugar cane cane juice less processed retains nutrients enjoy smooth rich dark chocolate health food guilt free moderation certainly enjoy eating rest chocolate bar watching dvdreally pleasantly surprised good granola not usually granola eater yes know supposed good whenever eaten past granola tasted dry tasting wow taste really good rolled oats nutty flavor enough spice okay evaporated cane juice second ingredient label sure sweet flavor really dark chocolate chunks really good flavor not bitter taste normally find dark chocolate freeze dried fruit nice sharp taste complaint comes play not many fruit chunks bag surprised chocolate chunks fruit chunks almost ratio not kill product nice contrast get fruit chunks nice would good young kids also like sour pop get fruit chunks munch would bonus serving size cup little small good eat calories per calories total g dietary gram proteinnothing ingredients worries pronounce words even oils sweetners okay book evaporated cane juice soy oil addition rolled oats also spelt flax seeds dried coconuts things would not eat otherwise another bonus non gmo verified happy bags good size oz servings per bag since get packs would think would take get unless eat every single day think price fair get would actually purchase family feel good eating must admit still shocked liking granolagranola crunch intended eaten like trail mix well balanced flavor ingredients chocolate chunks dried berries small easily mix handful coconut flavor quite pronounced perfect size storing desk drawer work somewhere climate controlled real chocolate could problem carry camping leave car hot day list ingredients shared natures path made love rolled oats evaporated cane juice soy oil rolled spelt wheat dark chocolate chunks evaporated cane juice unsweetened chocolate cocoa butter soy lecithin vanilla flax seeds dried coconut cocoa freeze dried berry blend freeze dried strawberries freeze dried raspberries rice starch sea salt natural chocolate flavor natural vanilla flavor tocopherols natural vitamin e organic produced facility uses dairy peanuts contains soy tree nuts whea tnature path dark chocolate red berries organic love crunch premium organic granola tastes good gets world granola talking real chocolate chunks strawberries raspberries retain flavor crisp oats ingredients listed package rolled oats evaporated cane juice soy oil rolled spelt wheat dark chocolate chunks evaporated cane juice unsweetened chocolate cocoa butter soy lecithin vanilla flax seeds dried coconut cocoa freeze dried berry blend freeze dried strawberries freeze dried raspberries rice starch sea salt natural chocolate flavor natural vanilla flavor tocopherols natural vitamin e usda certified organic contain no trans fat unfortunately must count calories no granola health food cup yummy granola contains calories fat much less interesting daily nosh post great grains raisins dates pecans calories cup fat calories yummy nature path calories per cup serving fat less yummy great grains puts nature path realm greatly enjoyed tasty treats mine rather daily staple five stars great flavor three stars fat high content average four starsnature path organic love crunch rolled oat granola little chunks dark chocolate bits freeze dried strawberries raspberries quality ingredients good tastes fresh quite crunchy primary criticism much sweet eat breakfast even turns milk sweet gives slight chocolate flavor granola much better eaten dry dessert would also good topping apple crisp suggested serving size cup calories would think normal serving would much larger possibly full cup calories half bag arrived tiny pieces shipping even suggested quarter cup would probably contain calories not low fat either cup suggested daily value fat note evaporated cane juice better known sugar ingredients rolled oats evaporated cane juice soy oil spelt dark chocolate chunks evaporated cane juice chocolate liquor cocoa butter soy lecithin vanilla flax seed dried coconut cocoa freeze dried berry blend freeze dried strawberries freeze dried raspberries rice starch sea salt natural chocolate flavor natural vanilla flavor tocopherols natural vitamin e nutrition facts serving size cup per container total package oz g per serving calories fat fat gsaturated fat fat gpolyunsaturated fat gmonounsaturated fat gcholesterol mgsodium mgpotassium mgtotal carbohydrates gdietary fiber gsugars gprotein gvitamin c'

In [87]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)
```

In [89]:

```
cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str]
```

In [24]:

```
def plot_WordCloud(txt, ind):

    """
    This function takes text as string and plots wordcloud. ind here is for the index of the list
element
    """
    stopwords = set(STOPWORDS)
    wordcloud = WordCloud(width = 1000, height = 600, background_color ='white', stopwords = stopwo
rds).generate(txt)
    # plot the WordCloud image
    plt.figure(figsize = (10, 8))
    plt.imshow(wordcloud, interpolation = 'bilinear')
    plt.axis("off")
    plt.title("World cloud of Cluster {}".format(ind))
    plt.tight_layout(pad = 0)

    plt.show()
```

In [99]:

```
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

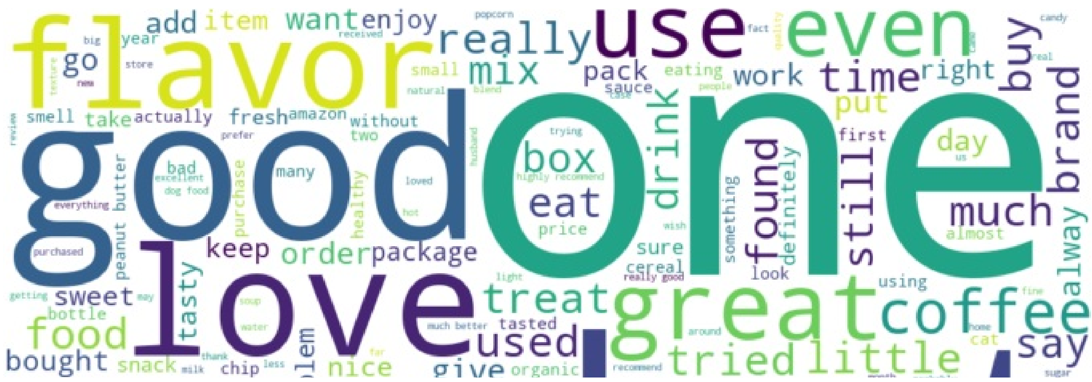
World cloud of Cluster 0

World cloud of Cluster 1

World cloud of Cluster 2

World cloud of Cluster 3

World cloud of Cluster 4



kmeans on TF-IDF

In [7]:

```
# ss
from sklearn.feature_extraction.text import TfidfVectorizer
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
tf_idf_vect.fit(X) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_tfidf = tf_idf_vect.transform(X)

print("After vectorizations")
print(X_train_tfidf.shape)
#print(X_cv_tfidf.shape, y_cv.shape)
print(type(X_train_tfidf))
```

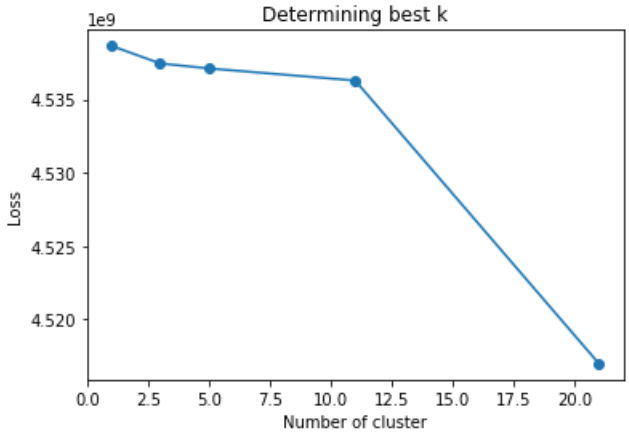
```
After vectorizations
(87773, 51709)
<class 'scipy.sparse.csr.csr_matrix'>
```

In [8]:

```
# Standardizing the tf-idf vectorizer
X_tfidf = StandardScaler(with_mean=False).fit_transform(X_train_tfidf)
```

In [10]:

```
getKmeansBestK(X_tfidf)
```



In [12]:

```
# Implementing k-means
kmeans_tfidf = KMeans(n_clusters= 11, n_jobs=-1, random_state= 507).fit(X_tfidf)
```

In [13]:

```
filename = 'kmeans_tfidf_model.sav'
pickle.dump(kmeans_tfidf, open(filename, 'wb'))
```

In [25]:

```
kmeans_tfidf.labels_
```

Out[25]:

```
array([3, 3, 3, ..., 3, 3, 3])
```

In [26]:

```
result = zip(X , kmeans_tfidf.labels_)
```

In [27]:

```
result_df = pd.DataFrame(result)
```

In [28]:

```
result_df.rename(columns={0: "CleanedText", 1: "tfidf_labels"}, inplace= True)
```

In [29]:

```
result_df.head()
```

Out[29]:

	CleanedText	tfidf_labels
0	bought apartment infested fruit flies hours tr...	3
1	really good idea final product outstanding use...	3
2	received shipment could hardly wait try produc...	3
3	nothing product bother link top page buy used ...	3
4	love stuff sugar free not rot gums tastes good...	3

In [30]:

```
# Number of reviews for each label
result_df['tfidf_labels'].value_counts()
```

Out[30]:

```
3     87008
1       415
6       170
9        74
8        38
5        35
7        17
2        11
10        2
0         2
4         1
Name: tfidf_labels, dtype: int64
```

In [32]:

```
# Based on the values of label column, we will insert the reviews to a separate list respectively.
# We then pass this list as str to wordcloud

reviews = result_df['CleanedText'].values

# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
```

```
cluster4 = []
cluster5 = []
cluster6 = []
cluster7 = []
cluster8 = []
cluster9 = []
cluster10 = []

for i in range(kmeans_tfidf.labels_.shape[0]):
    if kmeans_tfidf.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 3:
        cluster3.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 4:
        cluster4.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 5:
        cluster5.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 6:
        cluster6.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 7:
        cluster7.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 8:
        cluster8.append(reviews[i])

    elif kmeans_tfidf.labels_[i] == 9:
        cluster9.append(reviews[i])


    else :
        cluster10.append(reviews[i])
```

In [33]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)
cluster5_str = ('').join(cluster5)
cluster6_str = ('').join(cluster6)
cluster7_str = ('').join(cluster7)
cluster8_str = ('').join(cluster8)
cluster9_str = ('').join(cluster9)
cluster10_str = ('').join(cluster10)
```

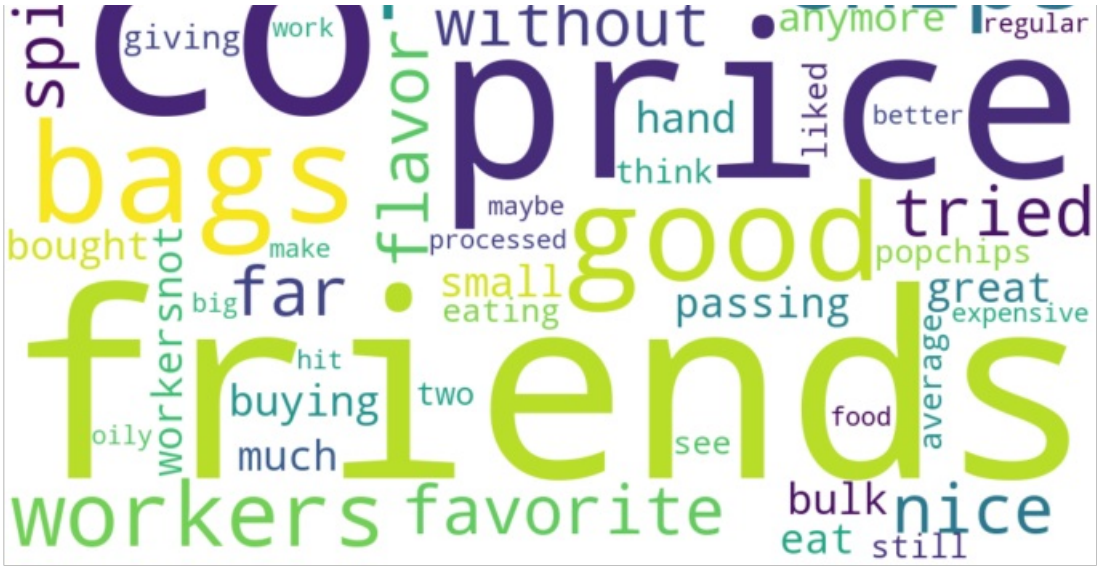
In [34]:

```
cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str, cluster5_str,
cluster6_str, cluster7_str, cluster8_str, cluster9_str, cluster10_str]
```

In [46]:

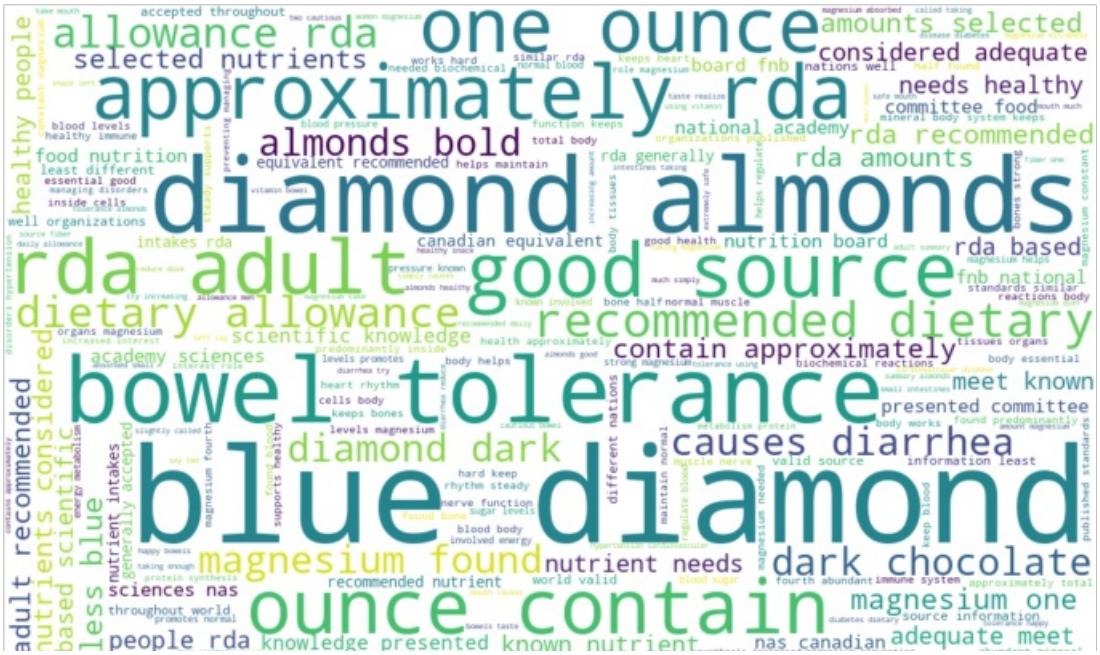
```
#fig, axes = plt.subplots(2, 6)
for ind, cluster in enumerate(cluster_str):
        plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0



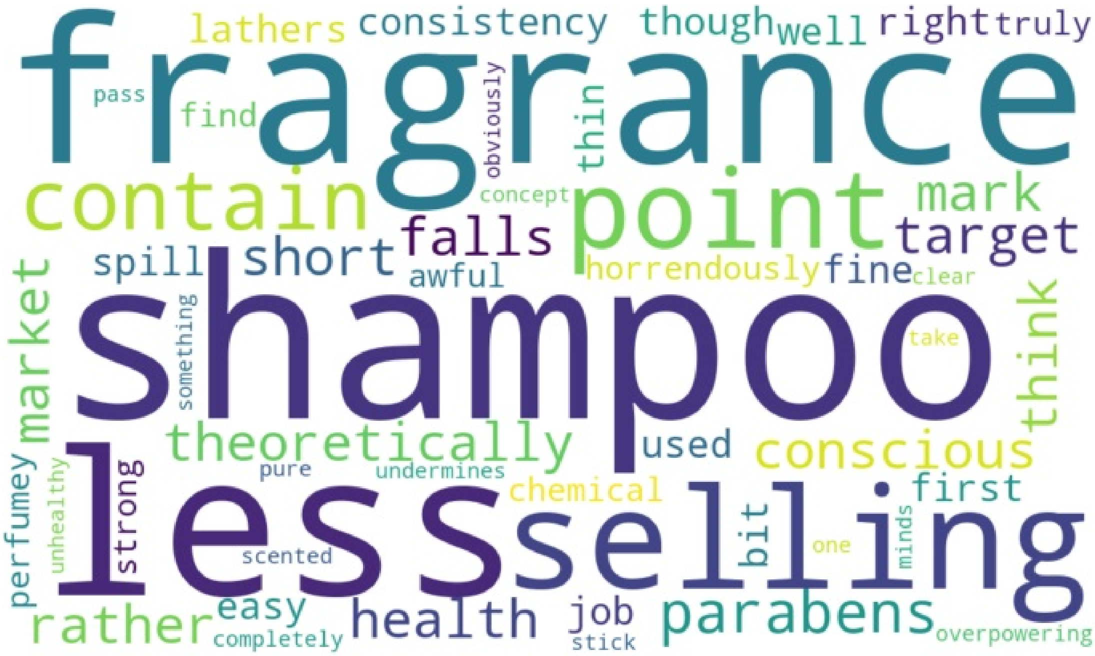
World cloud of Cluster 1

World cloud of Cluster 2

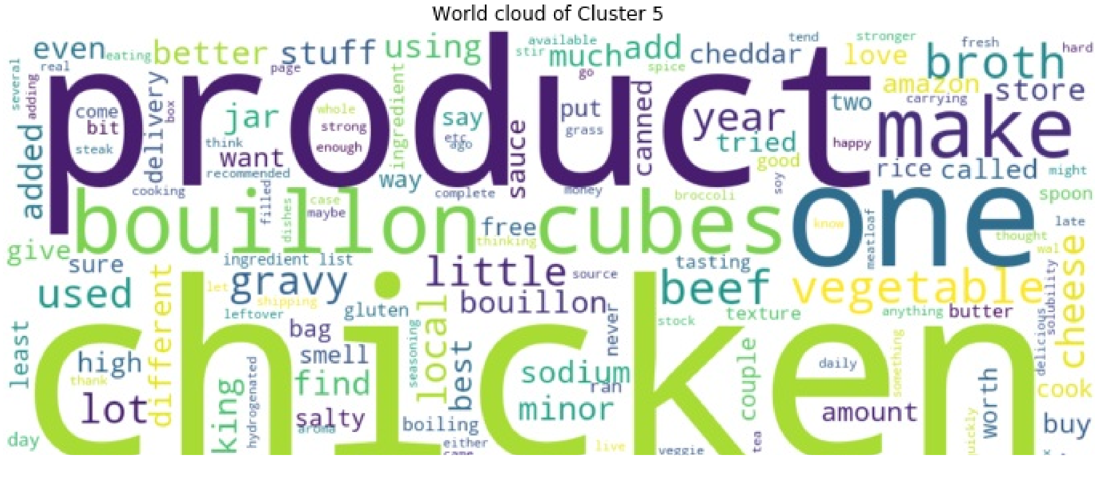


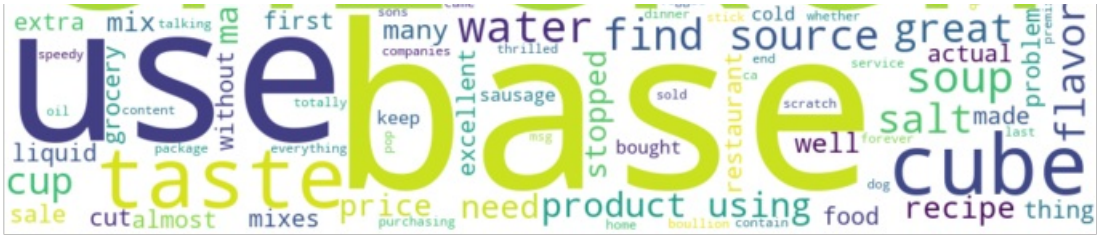
World cloud of Cluster 3

World cloud of Cluster 4

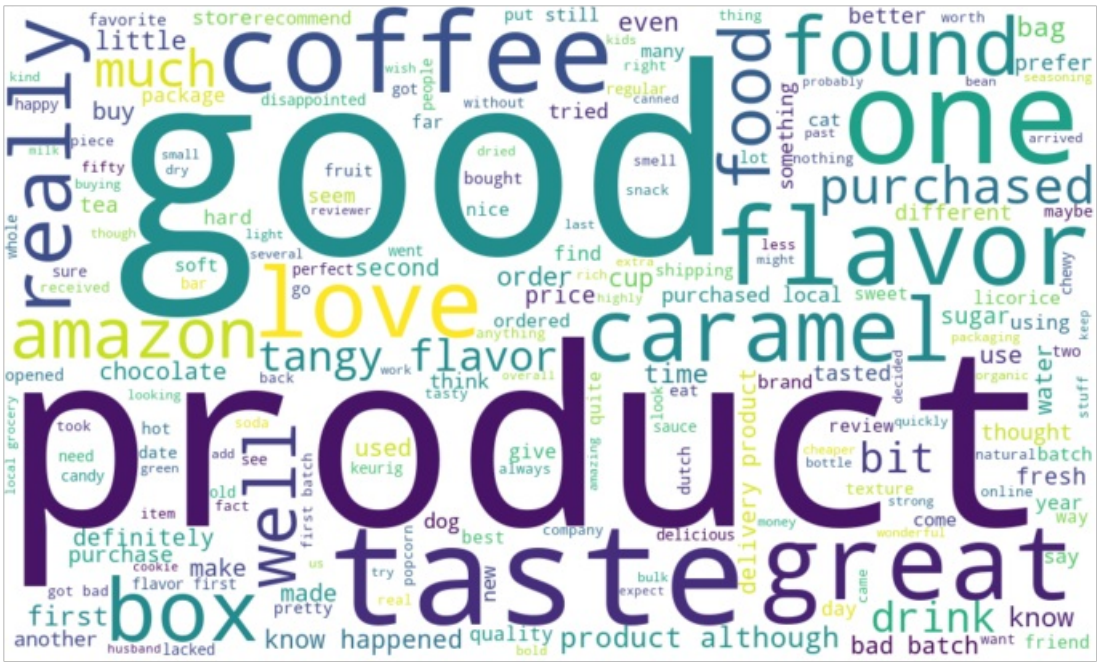


World cloud of Cluster 5



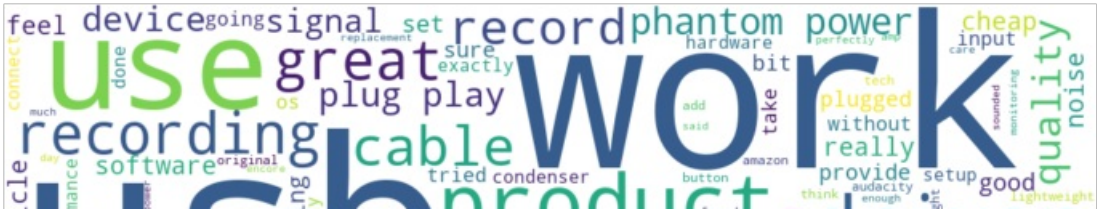


World cloud of Cluster 6



World cloud of Cluster 7

World cloud of Cluster 8



World cloud of Cluster 9

World cloud of Cluster 10

Word2vec

In [14]:

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

In [15]:

```
print(list_of_sentance_train[0])
```

```
['bought', 'apartment', 'infested', 'fruit', 'flies', 'hours', 'trap', 'attracted', 'many', 'flies
', 'within', 'days', 'practically', 'gone', 'may', 'not', 'long', 'term', 'solution', 'flies', 'dr
iving', 'crazy', 'consider', 'buying', 'one', 'caution', 'surface', 'sticky', 'try', 'avoid', 'tou
ching']
```

In [16]:

```
is_your_ram_gt_16g=False
want_to_use_google_w2v = False
want_to_train_w2v = True

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

elif want_to_use_google_w2v and is_your_ram_gt_16g:
    if os.path.isfile('GoogleNews-vectors-negative300.bin'):
        w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=Tr
ue)
        print(w2v_model.wv.most_similar('great'))
        print(w2v_model.wv.most_similar('worst'))
    else:
        print("you don't have gogole's word2vec file, keep want_to_train_w2v = True, to train your
own w2v ")
```

```
[('fantastic', 0.8441911935806274), ('good', 0.8308230042457581), ('excellent',
0.8168736696243286), ('awesome', 0.8125929236412048), ('perfect', 0.7970224618911743),
('terrific', 0.7934904098510742), ('wonderful', 0.7802051305770874), ('fabulous',
0.7312085032463074), ('amazing', 0.6994041800498962), ('decent', 0.6752279996871948)]
==================================================
[('greatest', 0.7632926106452942), ('best', 0.7325520515441895), ('tastiest', 0.7151054739952087),
('nastiest', 0.7017947435379028), ('coolest', 0.6454761028289795), ('disgusting',
0.6260499358177185), ('closest', 0.6239907741546631), ('nicest', 0.6225606203079224), ('horrible',
0.599621593952179), ('softest', 0.596433162689209)]
```

In [17]:

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

```
number of words that occured minimum 5 times  17386
sample words  ['bought', 'apartment', 'infested', 'fruit', 'flies', 'hours', 'trap', 'attracted',
'many', 'within', 'days', 'practically', 'gone', 'may', 'not', 'long', 'term', 'solution',
'driving', 'crazy', 'consider', 'buying', 'one', 'caution', 'surface', 'sticky', 'try', 'avoid', '
touching', 'really', 'good', 'idea', 'final', 'product', 'outstanding', 'use', 'car', 'window', 'e
verybody', 'asks', 'made', 'two', 'thumbs', 'received', 'shipment', 'could', 'hardly', 'wait', 'lo
ve', 'call']
```

In [18]:

```
# average Word2Vec
# compute average word2vec for each review.
sent_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
```

```
for sent in tqdm(list_of_sentance_train): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, you 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/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors_train.append(sent_vec)
sent_vectors_train = np.array(sent_vectors_train)
print(sent_vectors_train.shape)
print(sent_vectors_train[0])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 87773/87773 [02:54<00:00, 502.29it/s]
```

```
(87773, 50)
[-0.49829195  0.10107051 -0.58498146  0.02986002 -0.01582837 -0.00593825
 -0.11613219 -0.58249853  0.45320691 -0.08046174 -0.07771264 -0.2435316
  0.02650833  0.65270853 -0.30233298 -0.32471768  0.08576682 -0.10403201
 -0.28795816  0.15788848 -0.67258611 -0.12114539 -0.7946783  -0.17837132
  0.19039751 -0.06998545 -0.16292777 -0.28450224  0.2377823  -0.2231614
  0.38788906 -0.19385761 -0.49781666  0.08418349 -0.30370404  0.32890235
  0.35125909 -0.21964421 -0.11762911 -0.19953614 -0.20989574  0.38882576
 -0.35830842 -0.10261578  0.48089333  0.20616535 -0.37201625  0.26655469
 -0.02583727  0.63013505]
```

In [19]:

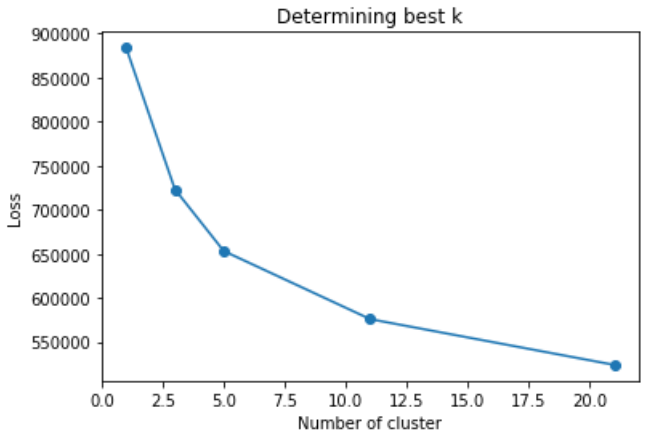
```
filename = 'kmeans_sent_vectors_train.sav'
pickle.dump(sent_vectors_train, open(filename, 'wb'))
```

In [6]:

```
sent_vectors_train = pickle.load(open('kmeans_sent_vectors_train.sav', 'rb'))
```

In [20]:

```
getKmeansBestK(sent_vectors_train)
```



Observation: For avg W2V, the best value of k is **5**

kmeans for avgw2v

In [21]:

```
# Implementing k-means
kmeans_avgw2v = KMeans(n_clusters= 5, n_jobs=-1, random_state= 507).fit(sent_vectors_train)
```

In [22]:

```
filename = 'kmeans_model_avgw2v.sav'
pickle.dump(kmeans_avgw2v, open(filename, 'wb'))
```

In [47]:

```
kmeans_avgw2v.labels_
```

Out[47]:

```
array([2, 1, 3, ..., 4, 4, 1])
```

In [48]:

```
result = zip(X , kmeans_avgw2v.labels_)
result_df = pd.DataFrame(result)
```

In [49]:

```
result_df.rename(columns={0: "CleanedText", 1: "avgw2v_labels"}, inplace= True)
result_df.head()
```

Out[49]:

	CleanedText	avgw2v_labels
0	bought apartment infested fruit flies hours tr...	2
1	really good idea final product outstanding use...	1
2	received shipment could hardly wait try produc...	3
3	nothing product bother link top page buy used ...	1
4	love stuff sugar free not rot gums tastes good...	0

In [51]:

```
# Number of reviews for each label
result_df['avgw2v_labels'].value_counts()
```

Out[51]:

```
1    23696
0    19132
3    18907
4    14396
2    11642
Name: avgw2v_labels, dtype: int64
```

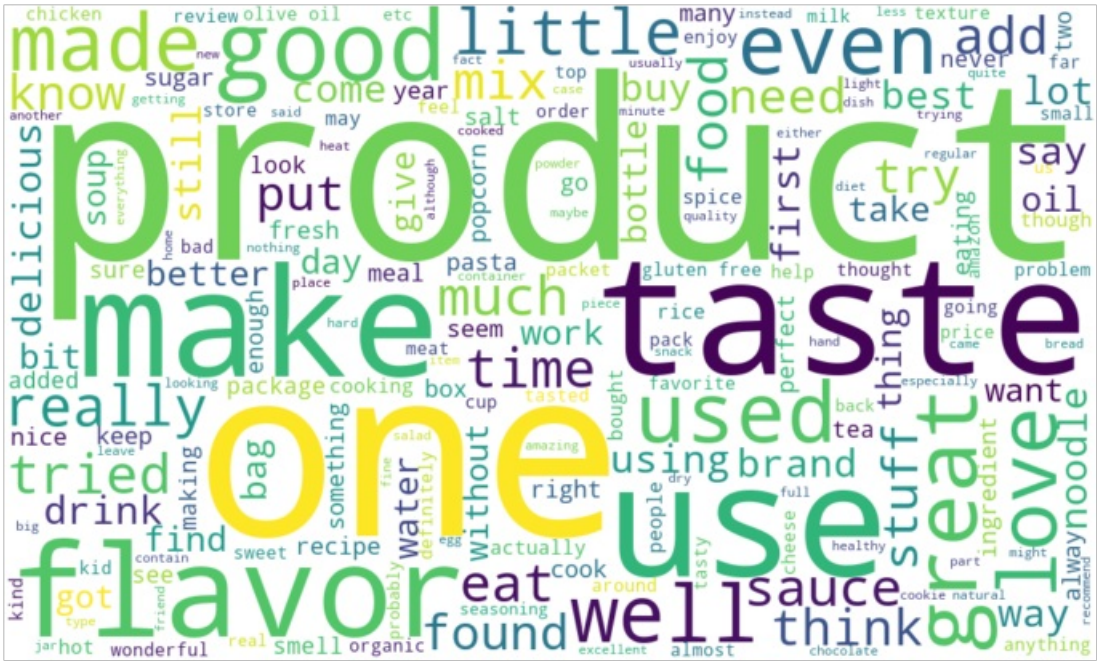
In [52]:

```
# Based on the values of label column, we will insert the reviews to a separate list respectively.
# We then pass this list as str to wordcloud

reviews = result_df['CleanedText'].values

# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []

for i in range(kmeans_avgw2v.labels_.shape[0]):
    if kmeans_avgw2v.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif kmeans_avgw2v.labels_[i] == 1:
        cluster1.append(reviews[i])
```

```
        cluster1.append(reviews[i])

    elif kmeans_avgw2v.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif kmeans_avgw2v.labels_[i] == 3:
        cluster3.append(reviews[i])

    else :
        cluster4.append(reviews[i])


cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)

cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str]

for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

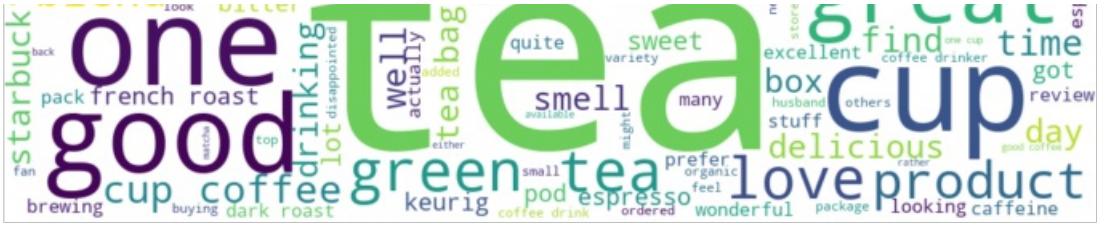
World cloud of Cluster 1



World cloud of Cluster 2

World cloud of Cluster 3

World cloud of Cluster 4



kmeans on TFIDF-weight word2vec

In [53]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
X_tfidfw2v = model.fit_transform(X)

# 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 [54]:

```
# TF-IDF weighted Word2Vec for sentences in X_train
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

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

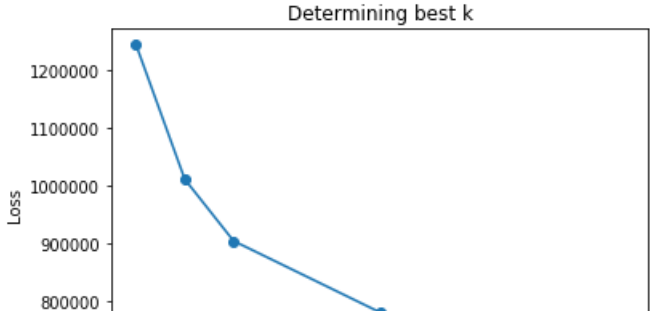
```
100%|██████████████████████████████████████████████████████████████████████████████| 87773/87773 [53
:48<00:00, 27.18it/s]
```

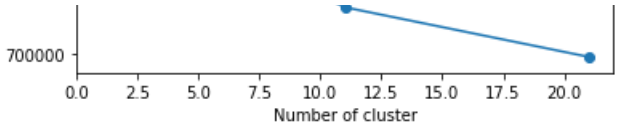
In [55]:

```
filename = 'kmeans_tfidf_sent_vectors_train.sav'
pickle.dump(tfidf_sent_vectors_train, open(filename, 'wb'))
```

In [56]:

```
getKmeansBestK(tfidf_sent_vectors_train)
```





Observation: From the above knee plot, we can see that n = 5 as there is a sudden dip

In [57]:

```
# Implementing k-means
kmeans_tfidfavgw2v = KMeans(n_clusters= 5, n_jobs=-1, random_state=
507).fit(tfidf_sent_vectors_train)
```

In [58]:

```
filename = 'kmeans_model_tfidfavgw2v.sav'
pickle.dump(kmeans_tfidfavgw2v, open(filename, 'wb'))
```

In [62]:

```
del result, result_df
```

In [63]:

```
kmeans_tfidfavgw2v.labels_
result = zip(X , kmeans_tfidfavgw2v.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "tfidfavgw2v_labels"}, inplace= True)
result_df.head()
```

Out[63]:

	CleanedText	tfidfavgw2v_labels
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	4

In [64]:

```
# Number of reviews for each label
result_df['tfidfavgw2v_labels'].value_counts()
```

Out[64]:

```
0    27734
4    21691
2    14669
3    12475
1    11204
Name: tfidfavgw2v_labels, dtype: int64
```

In [65]:

```
del reviews
```

In [66]:

```
# Based on the values of label column, we will insert the reviews to a separate list respectively.
# We then pass this list as str to wordcloud

reviews = result_df['CleanedText'].values
```

```
# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []

for i in range(kmeans_tfidfavgw2v.labels_.shape[0]):
    if kmeans_tfidfavgw2v.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif kmeans_tfidfavgw2v.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif kmeans_tfidfavgw2v.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif kmeans_tfidfavgw2v.labels_[i] == 3:
        cluster3.append(reviews[i])

    else :
        cluster4.append(reviews[i])
```

In [67]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)

cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str]
```

In [68]:

```
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1

World cloud of Cluster 2

World cloud of Cluster 3

Agglomerative Clustering

In [7]:

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

Set 3: Here, we will take 5k reviews of average word2vec reviews

In [9]:

```
# We load the already created word2vec vector
len(sent_vectors_train)
```

Out[9]:

```
87773
```

In [12]:

```
# From X, we only consider 5k records for agglomerative clustering
X_agglo = X[0:5000]
```

In [13]:

```
agglo_sent_vectors_train = sent_vectors_train[0:5000]
```

Agglomerative clustering for n = 2

In [14]:

```
clustering_avgw2v_n2 = AgglomerativeClustering(n_clusters=2).fit(agglo_sent_vectors_train)
```

In [15]:

```
clustering_avgw2v_n2.labels_
```

Out[15]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [16]:

```
result = zip(X_agglo , clustering_avgw2v_n2.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_avgw2v_labels_n2"}, inplace= True)
result_df.head()
```

Out[16]:

	CleanedText	agglo_avgw2v_labels_n2
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [17]:

```
result_df['agglo_avgw2v_labels_n2'].value_counts()
```

Out[17]:

```
0    4308
1     692
Name: agglo_avgw2v_labels_n2, dtype: int64
```

In [18]:

```
# Based on the values of label column, we will insert the reviews to a separate list respectively.
# We then pass this list as str to wordcloud

reviews = result_df['CleanedText'].values

# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []


for i in range(clustering_avgw2v_n2.labels_.shape[0]):
    if clustering_avgw2v_n2.labels_[i] == 0:
        cluster0.append(reviews[i])

    else :
        cluster1.append(reviews[i])
```

In [19]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
```

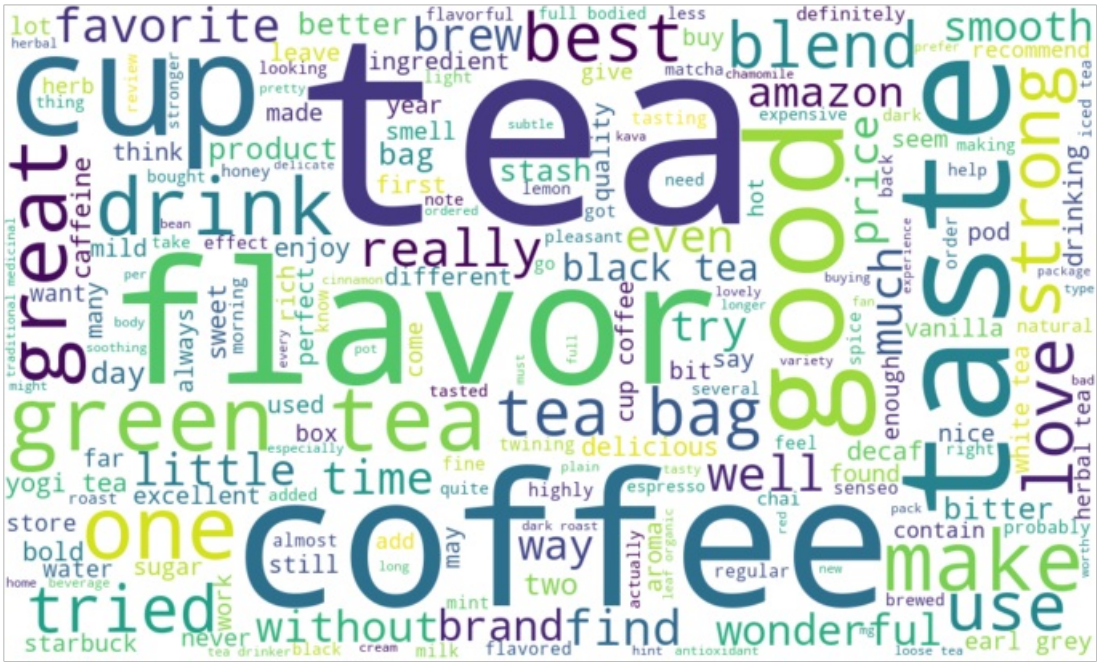
In [20]:

```
cluster_str = [cluster0_str, cluster1_str]
```

In [23]:

```
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 1



Description of cluster 0: This cluster is about eatables such cookies, bar and chocolates.

Description of cluster 1: This wordcloud is about tea and coffee and their tastes. The cluster is about amzon products and their positive reviews

Agglomerative Clustering for n = 3

In [24]:

```
clustering_avgw2v_n3 = AgglomerativeClustering(n_clusters=3).fit(agglo_sent_vectors_train)
```

In [25]:

```
result = zip(X_agglo , clustering_avgw2v_n3.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_avgw2v_labels_n3"}, inplace= True)
result_df.head()
```

Out[25]:

	CleanedText	agglo_avgw2v_labels_n3

	CleanedText	agglo_avgw2v_labels_n3
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	2

In [26]:

```
result_df['agglo_avgw2v_labels_n3'].value_counts()
```

Out[26]:

```
0    2296
2    2012
1     692
Name: agglo_avgw2v_labels_n3, dtype: int64
```

In [28]:

```
cluster0 = []
cluster1 = []
cluster2 = []

for i in range(clustering_avgw2v_n3.labels_.shape[0]):
    if clustering_avgw2v_n3.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif clustering_avgw2v_n3.labels_[i] == 1:
        cluster1.append(reviews[i])
    else :
        cluster2.append(reviews[i])
```

In [29]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
```

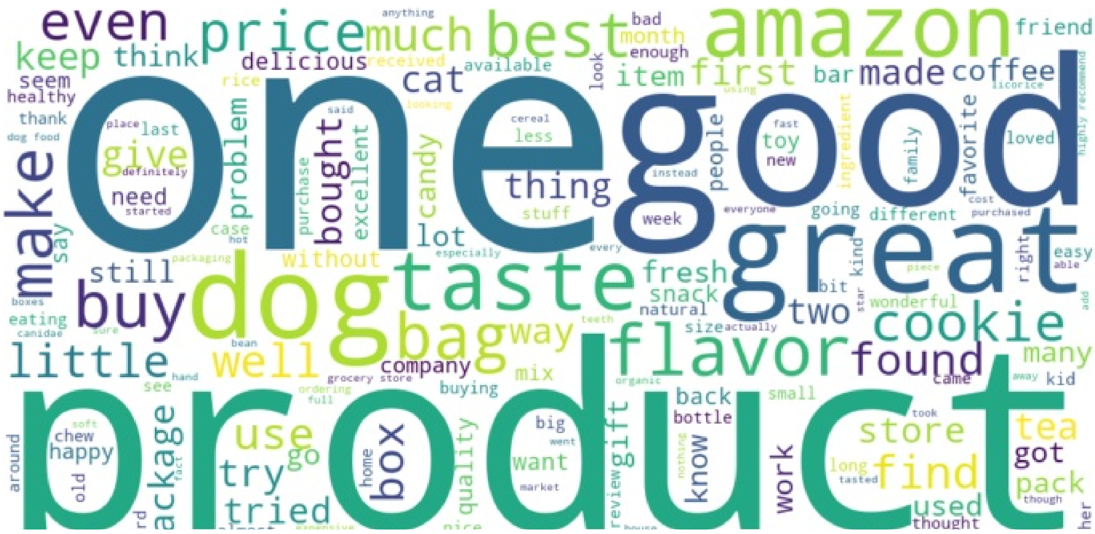
In [30]:

```
cluster_str = [cluster0_str, cluster1_str, cluster2_str]
```

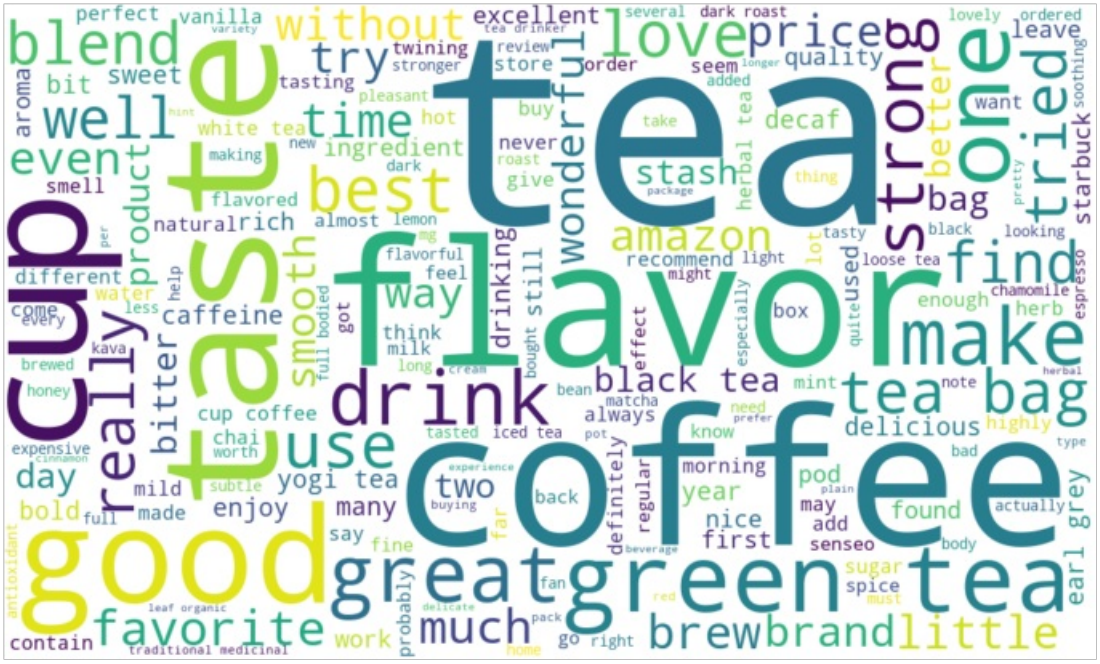
In [31]:

```
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0



World cloud of Cluster 1



World cloud of Cluster 2

Description of cluster 0: The cluster is about amazon products and their positive reviews

Description of cluster 1: This wordcloud is about tea and coffee and their tastes. The cluster is about amzon products and their positive reviews

Description of cluster 2: This cluster is about eatables such cookies, bar and chocolates.

Agglomerative Clustering for n = 5

In [32]:

```
clustering_avgw2v_n5 = AgglomerativeClustering(n_clusters=5).fit(agglo_sent_vectors_train)
```

In [33]:

```
result = zip(X_agglo , clustering_avgw2v_n5.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_avgw2v_labels_n5"}, inplace= True)
result_df.head()
```

Out[33]:

	CleanedText	agglo_avgw2v_labels_n5
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	2

In [35]:

```
result_df['agglo_avgw2v_labels_n5'].value_counts()
```

Out[35]:

```
0    1919
2    1179
4     833
1     692
3     377
Name: agglo_avgw2v_labels_n5, dtype: int64
```

In [37]:

```
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []

for i in range(clustering_avgw2v_n5.labels_.shape[0]):
    if clustering_avgw2v_n5.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif clustering_avgw2v_n5.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif clustering_avgw2v_n5.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif clustering_avgw2v_n5.labels_[i] == 3:
        cluster3.append(reviews[i])

    else :
        cluster4.append(reviews[i])
```

In [38]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)

cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str]

for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1

World cloud of Cluster 2

World cloud of Cluster 3

World cloud of Cluster 4

Description of cluster 0: The cluster is about amzon products and their positive reviews

Description of cluster 1: This wordcloud is about tea and coffee and their tastes. The cluster is about amzon products and their positive reviews

Description of cluster 2: This cluster is about eatables such cookies, bar and chocolates.

Description of cluster 3: This wordcloud is about dog food and their positive reviews.

Description of cluster 4: This wordcloud is about eatables such as rice, sugar etc and their tastes

Description of cluster 4: This workcloud is about eatables such as rice, sugar etc and their tastes

Agglomerative Clustering for n = 7

In [40]:

```
clustering_avgw2v_n7 = AgglomerativeClustering(n_clusters=7).fit(agglo_sent_vectors_train)
```

In [41]:

```
result = zip(X_agglo , clustering_avgw2v_n7.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_avgw2v_labels_n7"}, inplace= True)
result_df.head()
```

Out[41]:

	CleanedText	agglo_avgw2v_labels_n7
0	bought apartment infested fruit flies hours tr...	2
1	really good idea final product outstanding use...	2
2	received shipment could hardly wait try produc...	1
3	nothing product bother link top page buy used ...	2
4	love stuff sugar free not rot gums tastes good...	0

In [42]:

```
result_df['agglo_avgw2v_labels_n7'].value_counts()
```

Out[42]:

```
0    1179
2    1039
1     880
4     833
5     446
3     377
6     246
Name: agglo_avgw2v_labels_n7, dtype: int64
```

In [43]:

```
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []


for i in range(clustering_avgw2v_n7.labels_.shape[0]):
    if clustering_avgw2v_n7.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif clustering_avgw2v_n7.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif clustering_avgw2v_n7.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif clustering_avgw2v_n7.labels_[i] == 3:
        cluster3.append(reviews[i])

    elif clustering_avgw2v_n7.labels_[i] == 4:
        cluster4.append(reviews[i])

    elif clustering_avgw2v_n7.labels_[i] == 5:
        cluster5.append(reviews[i])
```

```
    else :
        cluster6.append(reviews[i])

cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)
cluster5_str = ('').join(cluster5)
cluster6_str = ('').join(cluster6)
```

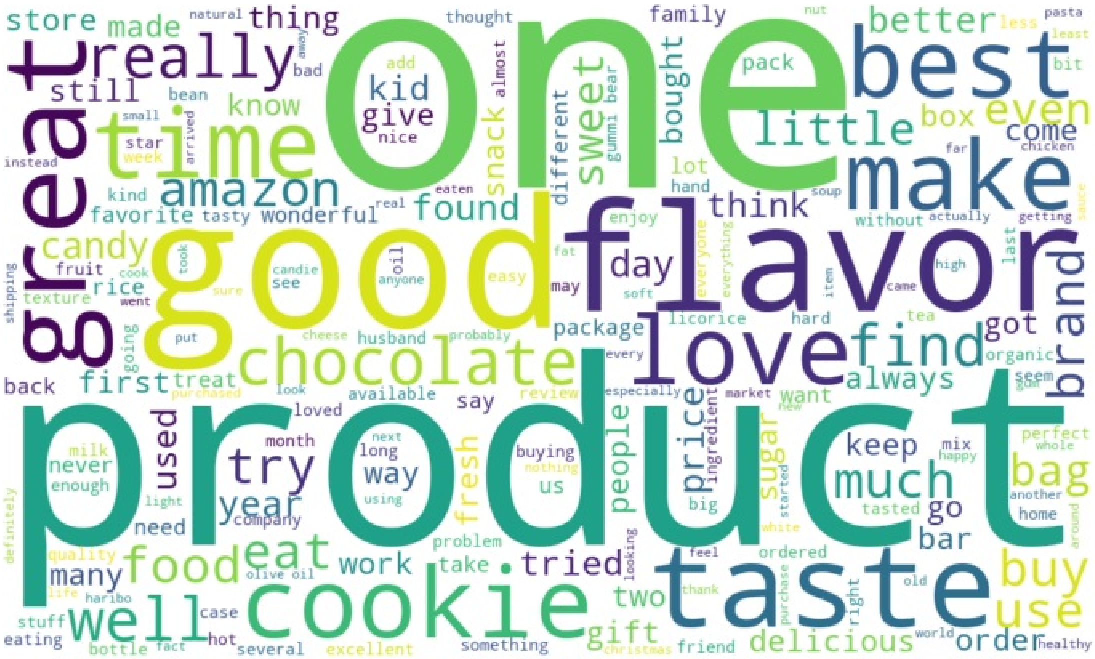
In [44]:

```
cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str, cluster5_str,
cluster6_str]

for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1

World cloud of Cluster 2



World cloud of Cluster 3

World cloud of Cluster 4

World cloud of Cluster 5

World cloud of Cluster 6

Description of cluster 0: This cluster is about eatables such cookies, bar and chocolates and their tastes

Description of cluster 1: The cluster is about amzon products and their positive reviews

Description of cluster 2: This cluster is about eatables such cookies, bar and chocolates.

Description of cluster 3: This wordcloud is about dog food and their positive reviews.

Description of cluster 4: This wordcloud is about eatables and their tastes

Description of cluster 5: This wordcloud is about tea and also about its variants such as green tea, organic, black tea etc

Description of cluster 5: This wordcloud is about tea and also about its variants such as green tea, organic, black tea etc

Description of cluster 6: This wordcloud is about flavor of coffee; decaf, bitter, smooth and mostly talks about all things coffee.

Set 4: Agglomerative clustering on TFIDF weighted word2vec

In [4]:

```
# We load the saved tfidf-weightedword2vec vector
tfidf_sent_vectors_train = pickle.load(open('kmeans_tfidf_sent_vectors_train.sav', 'rb'))
```

In [5]:

```
# We load the already created word2vec vector
len(tfidf_sent_vectors_train)
```

Out[5]:

```
87773
```

In [6]:

```
# From X, we only consider 5k records for agglomerative clustering
X_agglo = X[0:5000]
```

In [7]:

```
agglo_sent_vectors_train = tfidf_sent_vectors_train[0:5000]
```

Agglomerative clustering for n = 2

In [10]:

```
del clustering_avgw2v_n2
```

In [11]:

```
clustering_tfidf_avgw2v_n2 = AgglomerativeClustering(n_clusters=2).fit(agglo_sent_vectors_train)
```

In [12]:

```
result = zip(X_agglo , clustering_tfidf_avgw2v_n2.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_tfidf_avgw2v_labels_n2"}, inplace= True)
result_df.head()
```

Out[12]:

	CleanedText	agglo_tfidf_avgw2v_labels_n2
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [13]:

```
result_df['agglo_tfidf_avgw2v_labels_n2'].value_counts()
```

Out[13]:

```
0    3909
1    1091
Name: agglo_tfidf_avgw2v_labels_n2, dtype: int64
```

```
Name: agglo_tfidf_avgw2v_labels_n2, dtype: int64
```

In [14]:

```python
reviews = result_df['CleanedText'].values

# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []


for i in range(clustering_tfidf_avgw2v_n2.labels_.shape[0]):
    if clustering_tfidf_avgw2v_n2.labels_[i] == 0:
        cluster0.append(reviews[i])

    else :
        cluster1.append(reviews[i])
```

In [15]:

```python
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
```

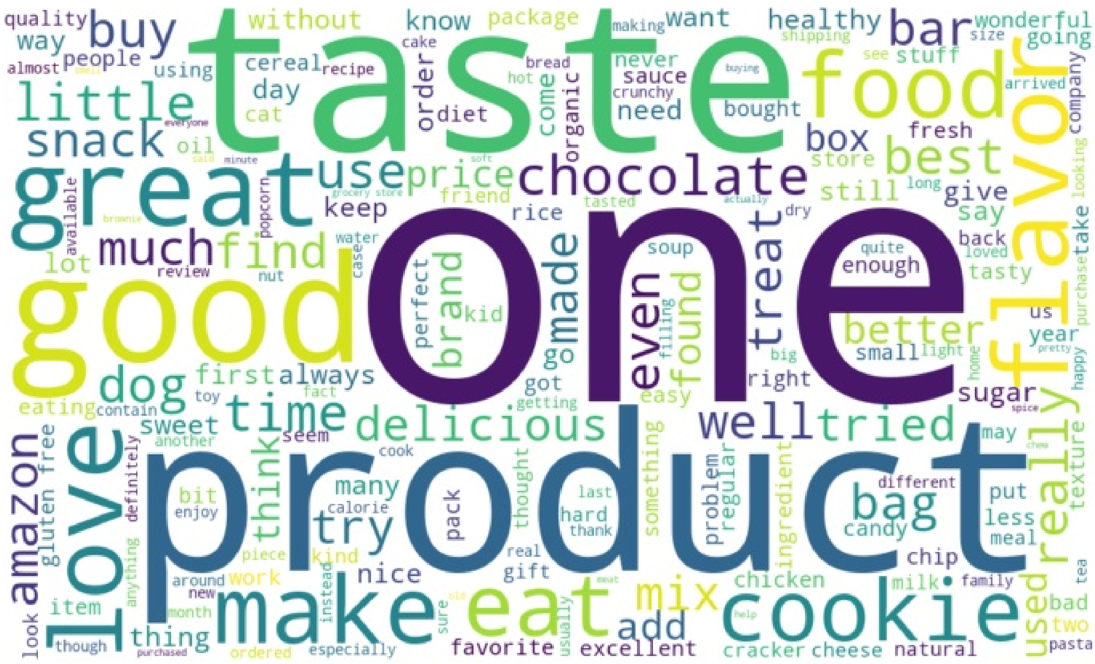
In [16]:

```python
cluster_str = [cluster0_str, cluster1_str]
```

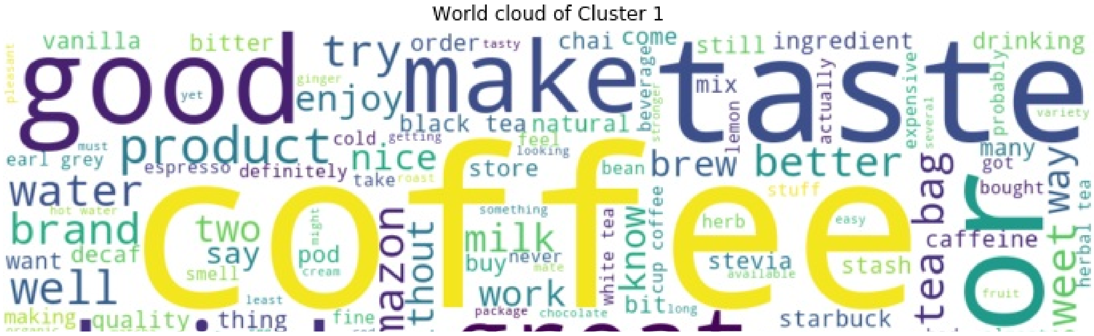
In [18]:

```python
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0



World cloud of Cluster 1



Description of cluster 0: This cluster is about eatables such cookies, bar and chocolates.

Description of cluster 1: The cluster is about beverages such as tea and coffee and their tastes.

Agglomerative Clustering for n = 3

In [19]:

```
clustering_tfidf_avgw2v_n3 = AgglomerativeClustering(n_clusters=3).fit(agglo_sent_vectors_train)
```

In [20]:

```
result = zip(X_agglo , clustering_tfidf_avgw2v_n3.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_tfidf_avgw2v_labels_n3"}, inplace= True)
result_df.head()
```

Out[20]:

	CleanedText	agglo_tfidf_avgw2v_labels_n3
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [21]:

```
result_df['agglo_tfidf_avgw2v_labels_n3'].value_counts()
```

Out[21]:

```
0    2024
2    1885
1    1091
Name: agglo_tfidf_avgw2v_labels_n3, dtype: int64
```

In [22]:

```
cluster0 = []
cluster1 = []
cluster2 = []

for i in range(clustering_tfidf_avgw2v_n3.labels_.shape[0]):
    if clustering_tfidf_avgw2v_n3.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif clustering_tfidf_avgw2v_n3.labels_[i] == 1:
        cluster1.append(reviews[i])
    else :
        cluster2.append(reviews[i])
```

In [23]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)

cluster_str = [cluster0_str, cluster1_str, cluster2_str]

for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1

World cloud of Cluster 2

Description of cluster 0: This wordcloud is about dog food and their positive reviews. This cluster is about eatables such cookies, bar and chocolates and their tastes

Description of cluster 1: The cluster is about tea and coffee and their tastes.

Description of cluster 2: This cluster is about eatables such cookies, bar and chocolates.

Agglomerative Clustering for n = 5

In [24]:

```
clustering_tfidf_avgw2v_n5 = AgglomerativeClustering(n_clusters=5).fit(agglo_sent_vectors_train)
```

In [25]:

```
result = zip(X_agglo , clustering_tfidf_avgw2v_n5.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_tfidf_avgw2v_labels_n5"}, inplace= True)
result_df.head()
```

Out[25]:

	CleanedText	agglo_tfidf_avgw2v_labels_n5
0	bought apartment infested fruit flies hours tr...	1
1	really good idea final product outstanding use...	1
2	received shipment could hardly wait try produc...	1
3	nothing product bother link top page buy used ...	1
4	love stuff sugar free not rot gums tastes good...	1

In [26]:

```
result_df['agglo_tfidf_avgw2v_labels_n5'].value_counts()
```

Out[26]:

```
0    1885
1    1677
2     923
3     347
4     168
Name: agglo_tfidf_avgw2v_labels_n5, dtype: int64
```

In [27]:

```
cluster0 = []
```

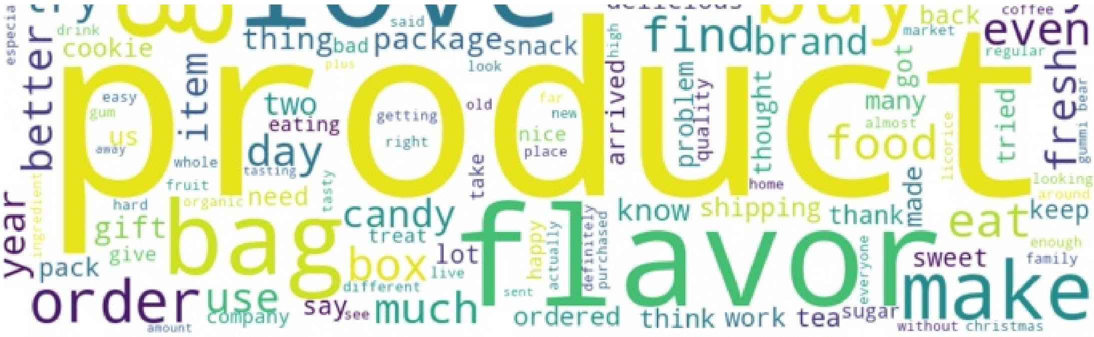
```
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []

for i in range(clustering_tfidf_avgw2v_n5.labels_.shape[0]):
    if clustering_tfidf_avgw2v_n5.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif clustering_tfidf_avgw2v_n5.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif clustering_tfidf_avgw2v_n5.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif clustering_tfidf_avgw2v_n5.labels_[i] == 3:
        cluster3.append(reviews[i])

    else :
        cluster4.append(reviews[i])
```

In [28]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)

cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str]

for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

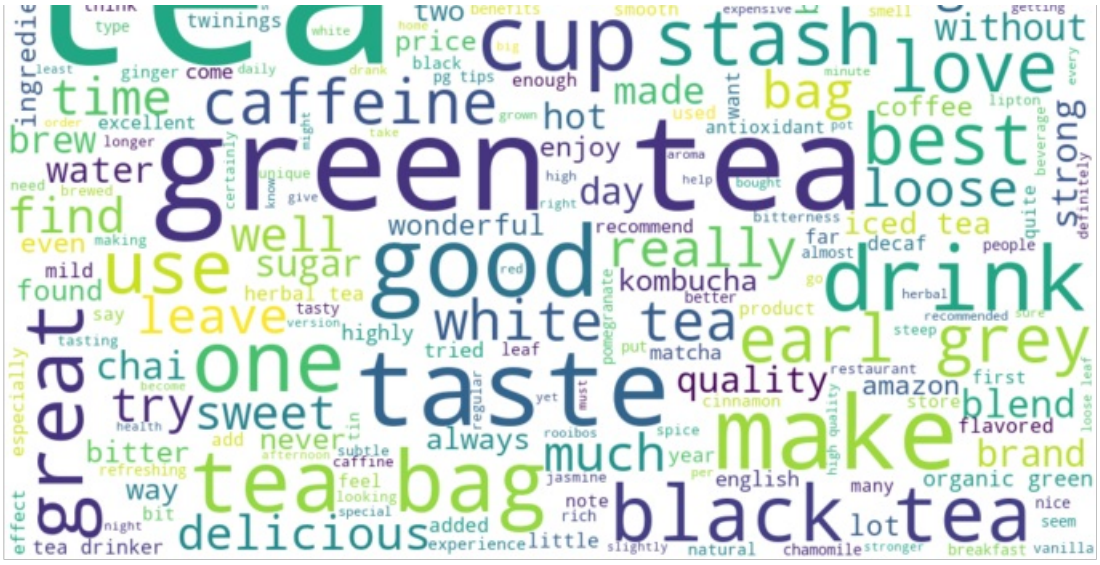
World cloud of Cluster 1



World cloud of Cluster 2

World cloud of Cluster 3

World cloud of Cluster 4



Description of cluster 0: This cluster is about eatables such cookies, bar and chocolates and their tastes

Description of cluster 1: The cluster is about products and their positive reviews

Description of cluster 2: This cluster is about eatables such cookies, bar and chocolates.

Description of cluster 3: This wordcloud is about dog and cat food and their positive reviews.

Description of cluster 4: This wordcloud is about tea and also about its variants such as green tea, organic, black tea etc

Agglomerative Clustering for n = 7

In [29]:

```
clustering_tfidf_avgw2v_n7 = AgglomerativeClustering(n_clusters=7).fit(agglo_sent_vectors_train)
```

In [30]:

```
result = zip(X_agglo , clustering_tfidf_avgw2v_n7.labels_)
result_df = pd.DataFrame(result)
result_df.rename(columns={0: "CleanedText", 1: "agglo_tfidf_avgw2v_labels_n7"}, inplace= True)
result_df.head()
```

Out[30]:

	CleanedText	agglo_tfidf_avgw2v_labels_n7
0	bought apartment infested fruit flies hours tr...	1
1	really good idea final product outstanding use...	1
2	received shipment could hardly wait try produc...	1
3	nothing product bother link top page buy used ...	1
4	love stuff sugar free not rot gums tastes good...	1

In [31]:

```
result_df['agglo_tfidf_avgw2v_labels_n7'].value_counts()
```

Out[31]:

```
1    1677
2    1183
0     702
5     636
3     347
6     287
4     168
```

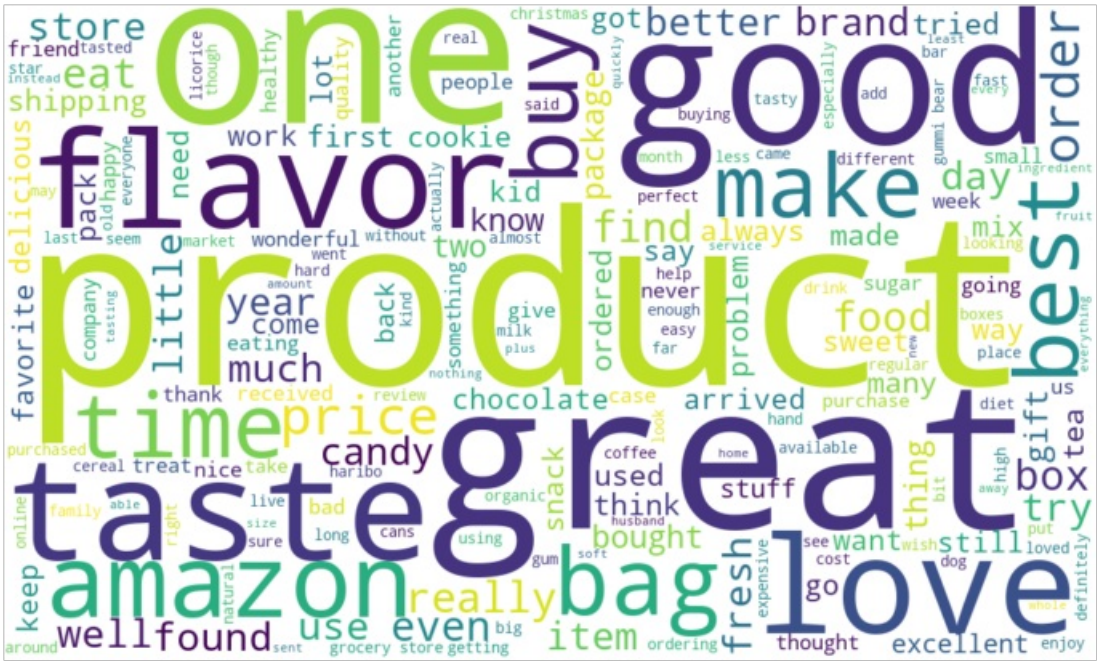
```
Name: agglo_tfidf_avgw2v_labels_n7, dtype: int64
```

In [32]:

```
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []


for i in range(clustering_tfidf_avgw2v_n7.labels_.shape[0]):
    if clustering_tfidf_avgw2v_n7.labels_[i] == 0:
        cluster0.append(reviews[i])

    elif clustering_tfidf_avgw2v_n7.labels_[i] == 1:
        cluster1.append(reviews[i])

    elif clustering_tfidf_avgw2v_n7.labels_[i] == 2:
        cluster2.append(reviews[i])

    elif clustering_tfidf_avgw2v_n7.labels_[i] == 3:
        cluster3.append(reviews[i])

    elif clustering_tfidf_avgw2v_n7.labels_[i] == 4:
        cluster4.append(reviews[i])

    elif clustering_tfidf_avgw2v_n7.labels_[i] == 5:
        cluster5.append(reviews[i])

    else :
        cluster6.append(reviews[i])

cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
cluster2_str = ('').join(cluster2)
cluster3_str = ('').join(cluster3)
cluster4_str = ('').join(cluster4)
cluster5_str = ('').join(cluster5)
cluster6_str = ('').join(cluster6)

cluster_str = [cluster0_str, cluster1_str, cluster2_str, cluster3_str, cluster4_str, cluster5_str,
cluster6_str]

for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1



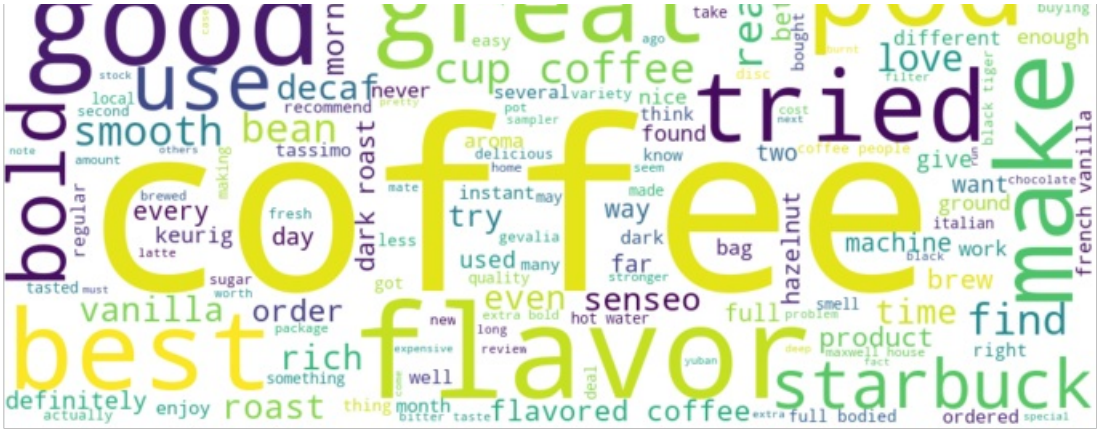
World cloud of Cluster 2

World cloud of Cluster 3

World cloud of Cluster 4

World cloud of Cluster 5

World cloud of Cluster 6



Description of cluster 0: Its about flavor and taste. There are other positive words such as good, great etc

Description of cluster 1: The cluster is about products and their positive reviews

Description of cluster 2: This cluster is about eatables such cookies, bar and chocolates.

Description of cluster 3: This wordcloud is about dog food and their positive reviews.

Description of cluster 4: This wordcloud is about tea and also about its variants such as green tea, organic, black tea etc

Description of cluster 5: This wordcloud is about coffee and its flavor.

Description of cluster 6: This wordcloud is about flavor of coffee, places like starbuck, senseo and mostly talks about all things coffee.

DBSCAN Algorithm

DBSCAN on average word2vec

In [2]:

```
# loading pickle file for average word2vec
sent_vectors_train = pickle.load(open('kmeans_sent_vectors_train.sav', 'rb'))
```

In [3]:

```
sent_vectors_train.shape[1]
```

Out[3]:

```
50
```

In [4]:

```
sent_vectors_train_5k = sent_vectors_train[0:5000] # we only select 5k points
```

Note: As the rule of thumb, we take minPts value as 2 *dimensionality. So we will set minPts* = 250=100

In [5]:

```
minPts = 100
```

In [7]:

```
# reference https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r/4855
8030#48558030
from sklearn.neighbors import NearestNeighbors
```

In [8]:

```
knn_clf = NearestNeighbors(n_neighbors= minPts)
```

In [9]:

```
knn_clf.fit(sent_vectors_train_5k)
```

Out[9]:

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

In [12]:

```
distances,  indices = knn_clf.kneighbors()
```

In [13]:

```
len(distances)
```

Out[13]:

```
5000
```

In [14]:

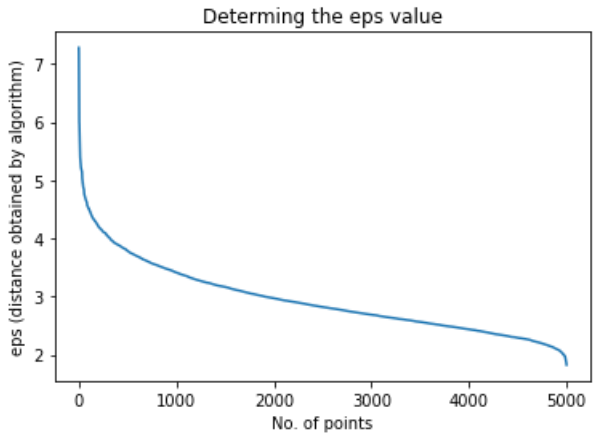
```
distanceDesc = sorted(distances[:, minPts-1], reverse = True)
```

In [19]:

```
plt.plot(list(range(1, 5000+1)), distanceDesc)
plt.xlabel('No. of points')
plt.ylabel("eps (distance obtained by algorithm)");
plt.title('Determing the eps value')
```

Out[19]:

```
Text(0.5, 1.0, 'Determing the eps value')
```



Observation: We can observe a knee point at 4. Therefore, we will consider eps value to be 4

In [20]:

```
eps = 4
```

In [23]:

```
# we only select 5k points
X_dbscan = X[0:5000]
```

In [28]:

```
dbscan_avgw2v= DBSCAN(eps = 4, min_samples= 100)
```

In [29]:

```
dbscan_avgw2v.fit(sent_vectors_train_5k)
```

Out[29]:

```
DBSCAN(algorithm='auto', eps=4, leaf_size=30, metric='euclidean',
    metric_params=None, min_samples=100, n_jobs=None, p=None)
```

In [30]:

```
filename = 'dbscan_avgw2v_model.sav'
pickle.dump(dbscan_avgw2v, open(filename, 'wb'))
```

In [31]:

```
result = zip(X_dbscan , dbscan_avgw2v.labels_)
```

In [32]:

```
result_df = pd.DataFrame(result)
result_df.head()
```

Out[32]:

	0	1
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [33]:

```
result_df.rename(columns={0: "CleanedText", 1: "avgw2v_labels"}, inplace= True)
result_df.head()
```

Out[33]:

	CleanedText	avgw2v_labels
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [34]:

```
# Number of reviews for each label
result_df['avgw2v_labels'].value_counts()
```

Out[34]:

```
 0    4982
-1      18
Name: avgw2v_labels, dtype: int64
```

In [35]:

```
# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []
```

In [37]:

```
reviews = result_df['CleanedText'].values
```

In [38]:

```
for i in range(dbscan_avgw2v.labels_.shape[0]):
    if dbscan_avgw2v.labels_[i] == 0:
        cluster0.append(reviews[i])

    else :
        cluster1.append(reviews[i])
```

In [39]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
```

In [40]:

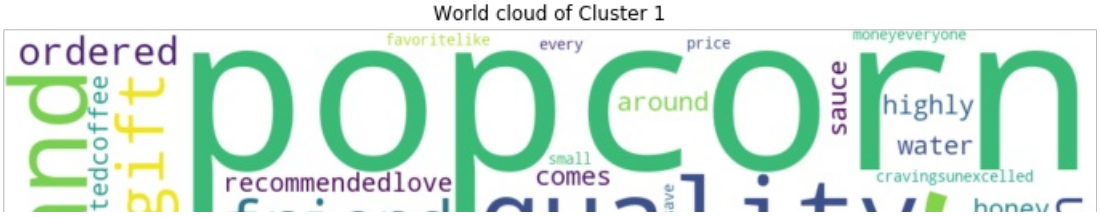
```
cluster_str = [cluster0_str, cluster1_str]
```

In [41]:

```
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1



Description of word cloud 0: The cluster is about beverages, snacks other consumable products and their reviews seems to be positive

Description of word cloud 1: This cluster is about snacks and their tastes.

DBSCAN on TFIDF-weighted word2vec

In [47]:

```
# loading pickle file for tfidf weighted word2vec
tfidf_weightw2v_dbscan = pickle.load(open('kmeans_tfidf_sent_vectors_train.sav', 'rb'))
```

In [51]:

```
len(tfidf_weightw2v_dbscan)
```

Out[51]:

87773

Note: As the rule of thumb, we take minPts value as 2 *dimensionality. So we will set minPts* = 2*50=100

In [50]:

```
tfidf_weightw2v_dbscan_5k = tfidf_weightw2v_dbscan[0:5000] # we only select 5k points
```

In [52]:

```
minPts = 100
```

In [53]:

```
# reference https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r/48558030#48558030
from sklearn.neighbors import NearestNeighbors
```

In [54]:

```
knn_clf = NearestNeighbors(n_neighbors= minPts)
```

In [55]:

```
knn_clf.fit(tfidf_weightw2v_dbscan_5k)
```

Out[55]:

```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='minkowski',
```

```
        metric_params=None, n_jobs=None, n_neighbors=100, p=2, radius=1.0)
```

In [56]:

```
distances,  indices = knn_clf.kneighbors()
```

In [57]:

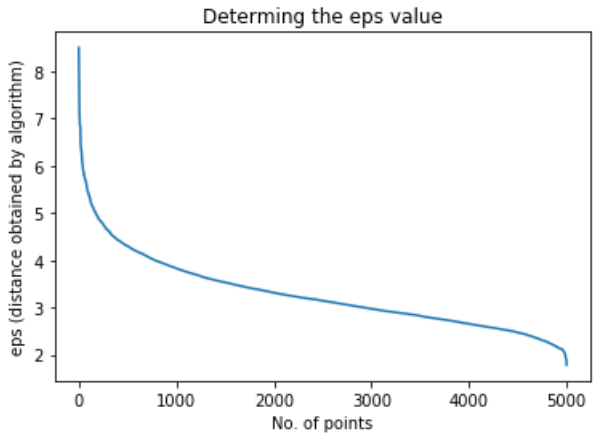
```
distanceDesc = sorted(distances[:, minPts-1], reverse = True)
```

In [58]:

```
plt.plot(list(range(1, 5000+1)), distanceDesc)
plt.xlabel('No. of points')
plt.ylabel("eps (distance obtained by algorithm)");
plt.title('Determing the eps value')
```

Out[58]:

```
Text(0.5, 1.0, 'Determing the eps value')
```



Observation: We can observe a knee point at 5. Therefore, we will consider eps value to be 5

In [59]:

```
dbscan_tfidf_weightedw2v= DBSCAN(eps = 4.5, min_samples= 100)
```

In [60]:

```
dbscan_tfidf_weightedw2v.fit(tfidf_weightw2v_dbscan_5k)
```

Out[60]:

```
DBSCAN(algorithm='auto', eps=5, leaf_size=30, metric='euclidean',
    metric_params=None, min_samples=100, n_jobs=None, p=None)
```

In [61]:

```
filename = 'dbscan_tfidf_weighted2v_model.sav'
pickle.dump(dbscan_tfidf_weightedw2v, open(filename, 'wb'))
```

In [62]:

```
result = zip(X_dbscan , dbscan_tfidf_weightedw2v.labels_)
```

In [63]:

```
result_df = pd.DataFrame(result)
result_df.head()
```

Out[63]:

	0	1
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [64]:

```
result_df.rename(columns={0: "CleanedText", 1: "tfidf_weightedw2v_labels"}, inplace= True)
result_df.head()
```

Out[64]:

	CleanedText	tfidf_weightedw2v_labels
0	bought apartment infested fruit flies hours tr...	0
1	really good idea final product outstanding use...	0
2	received shipment could hardly wait try produc...	0
3	nothing product bother link top page buy used ...	0
4	love stuff sugar free not rot gums tastes good...	0

In [65]:

```
# Number of reviews for each label
result_df['tfidf_weightedw2v_labels'].value_counts()
```

Out[65]:

```
 0    4991
-1       9
Name: tfidf_weightedw2v_labels, dtype: int64
```

In [66]:

```
# Creating empty lists to store reviews based on the label
cluster0 = []
cluster1 = []
```

In [67]:

```
reviews = result_df['CleanedText'].values
```

In [68]:

```
for i in range(dbscan_tfidf_weightedw2v.labels_.shape[0]):
    if dbscan_tfidf_weightedw2v.labels_[i] == 0:
        cluster0.append(reviews[i])

    else :
        cluster1.append(reviews[i])
```

In [69]:

```
cluster0_str = ('').join(cluster0)
cluster1_str = ('').join(cluster1)
```

In [70]:

```
cluster_str = [cluster0_str, cluster1_str]
```

In [71]:

```
for ind, cluster in enumerate(cluster_str):
    plot_WordCloud(cluster, ind)
```

World cloud of Cluster 0

World cloud of Cluster 1

Description of word cloud 0: The cluster is about beverages, snacks other consumable products and their reviews seems to be positive

Description of word cloud 1: This cluster talks about supplements like amino acids, vitamins etc