Kmeans , Agglomerative , DBSCAN

In [1]:

```
#to ignore warnings
import warnings
warnings.filterwarnings("ignore")
#to use sqlite3 database
import sqlite3
import numpy as np
import pandas as pd
import string
import nltk
import matplotlib.pyplot as plt

from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import cross_validation
from sklearn.cross_validation import cross_val_score
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
C:\Users\krush\Anaconda3\lib\site-packages\sklearn\cross_validation.py:
41: DeprecationWarning: This module was deprecated in version 0.18 in f
avor of the model_selection module into which all the refactored classe
s and functions are moved. Also note that the interface of the new CV i
terators are different from that of this module. This module will be re
moved in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

PREPROCESSING

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

In [2]:
```
con = sqlite3.connect('database.sqlite')

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


# Give reviews with Score>3 a positive rating, and reviews with a score
<3 a negative rating.
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
```

In [3]:
```
stop = set(stopwords.words('english')) #set of stopwords
sno = nltk.stem.SnowballStemmer('english') #initialising the snowball s
temmer

def cleanhtml(sentence): #function to clean the word of any html-tags
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', sentence)
    return cleantext
def cleanpunc(sentence): #function to clean the word of any punctuation
 or special characters
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    return  cleaned
```

In [4]:
```
#Code for implementing step-by-step the checks mentioned in the pre-pro
cessing phase
# this code takes a while to run as it needs to run on 500k sentences.
i=0
str1=' '
```

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

```
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
s=''
for sent in filtered_data['Text'].values:
    filtered_sentence=[]
    #print(sent);
    sent=cleanhtml(sent) # remove HTMl tags
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(sno.stem(cleaned_words.lower())).encode('utf8')
                    filtered_sentence.append(s)
                    if (filtered_data['Score'].values)[i] == 'positive'
:
                        all_positive_words.append(s) #list of all words
 used to describe positive reviews
                    if(filtered_data['Score'].values)[i] == 'negative':
                        all_negative_words.append(s) #list of all words
 used to describe negative reviews reviews
                else:
                    continue
            else:
                continue
    #print(filtered_sentence)
    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    #print("***********************************************************
************")

    final_string.append(str1)
    i+=1
```

In [5]:
```
filtered_data['CleanedText']=final_string #adding a column of CleanedTe
xt which displays the data after pre-processing of the review
filtered_data['CleanedText']=filtered_data['CleanedText'].str.decode("u
tf-8")
```

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

KMEANS FOR BOW, TF-IDF , AVG W2V, TFIDF-W2V

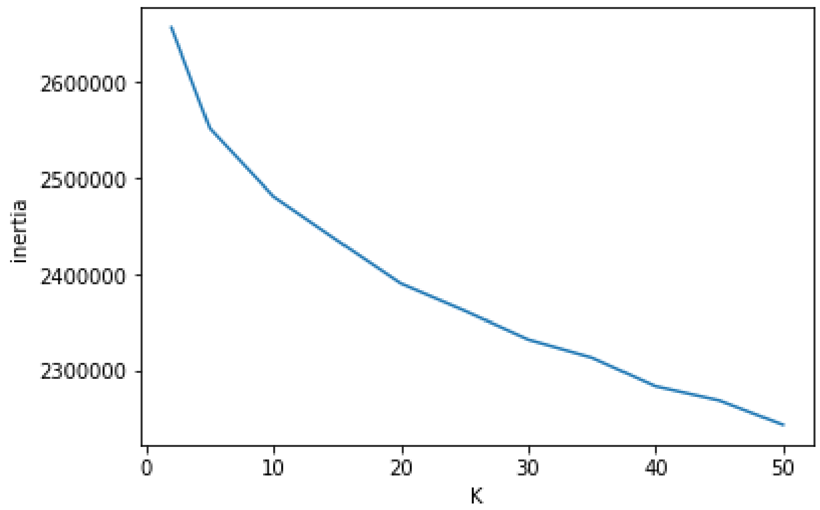
In [6]:
```
sorted_data=filtered_data.sort_values(by=['Time'])
sampledata = sorted_data.head(50000)
```

In [7]:
```
count_vect = CountVectorizer(min_df=10) #in scikit-learn
vec = count_vect.fit(sampledata['CleanedText'].values)
X_vec = vec.transform(sampledata['CleanedText'].values)
```

In [24]:
```
from sklearn.cluster import KMeans
K = [2,5,10,15,20,25,30,35,40,45,50]
inertia = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0).fit(X_vec)
    inertia.append(kmeans.inertia_)
```

In [40]:
```
plt.plot(K,inertia)
plt.xlabel('K')
plt.ylabel('inertia')
plt.show()
```

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



In [8]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=20, random_state=0).fit(X_vec)
```

In [9]:
```
kmeans.labels_
```

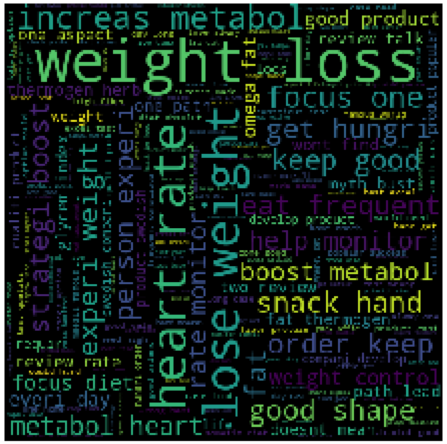
Out[9]:
```
array([ 9,  9,  9, ..., 18,  6,  9])
```

In [11]:
```
def againcleaning(X):
    comment_words=' '
    for words in X:
        comment_words = comment_words + words + ' '
    return comment_words
count=0
review=sampledata['CleanedText'].values
topn_class1 = sorted(zip(kmeans.labels_, review))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
```

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

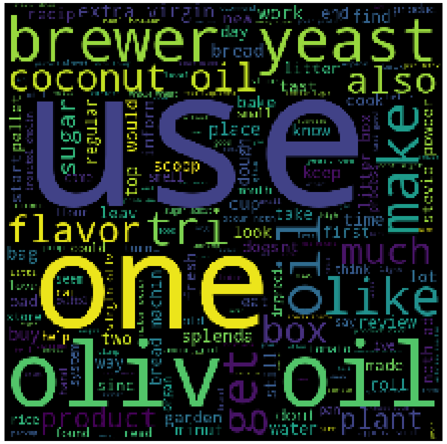
```
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=500,height=500).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=500,
height=500).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = 0
```

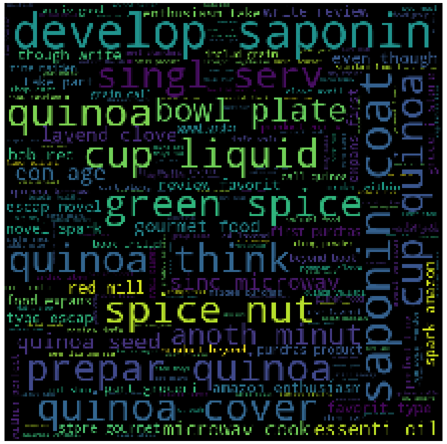



```
cluster = 1
```

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

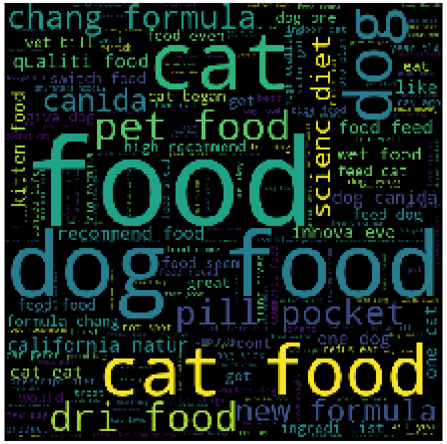


cluster = 2

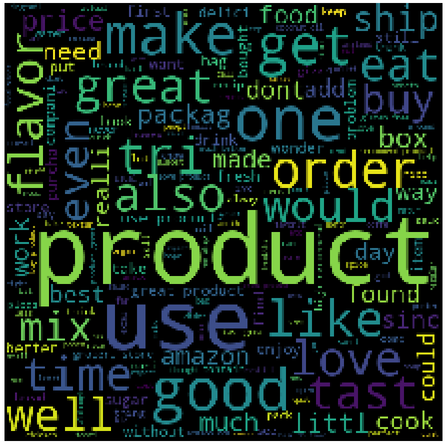


cluster = 3

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

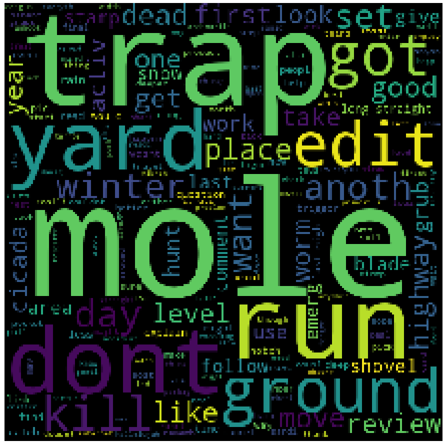


cluster = 4



cluster = 5

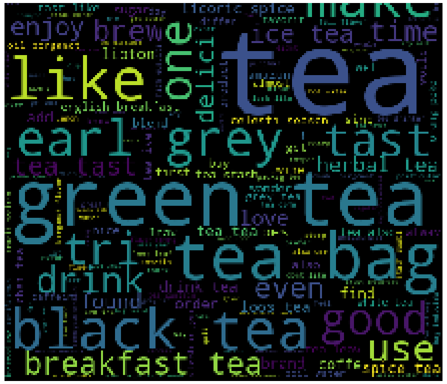
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



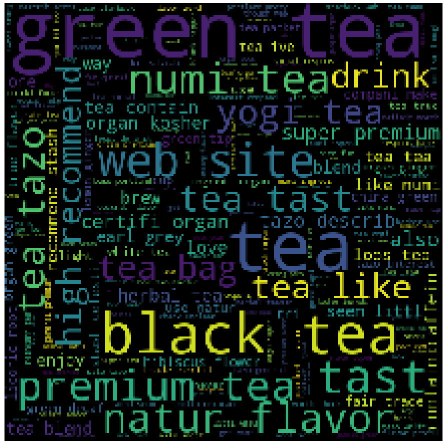
cluster = 6

cluster = 7

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

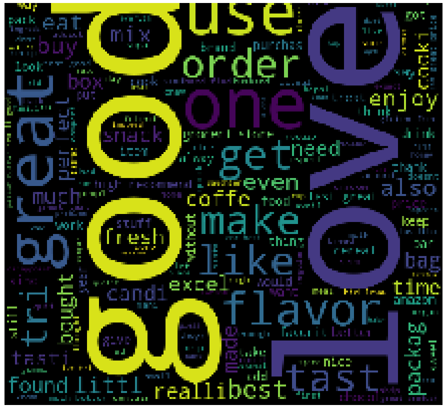


cluster = 8

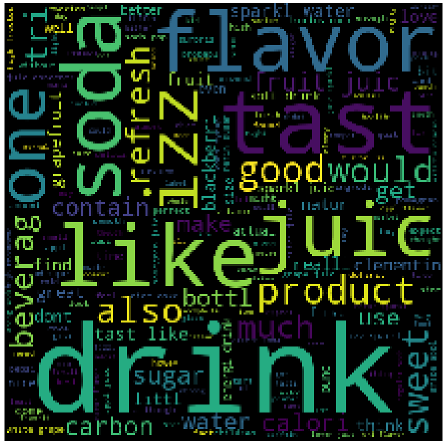


cluster = 9

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



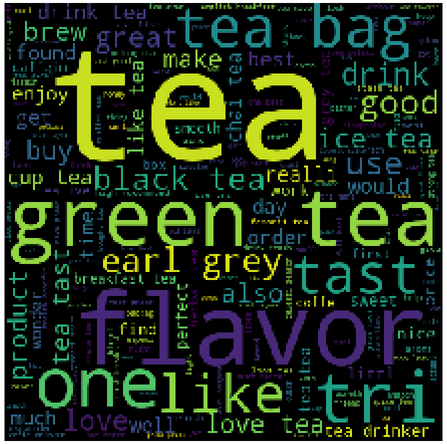
cluster = 10



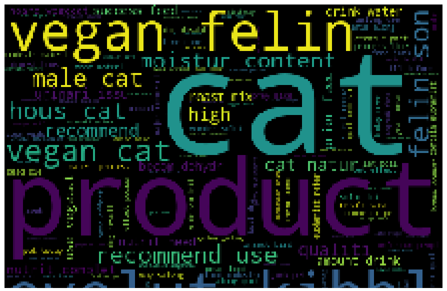
cluster = 11

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

cluster = 12

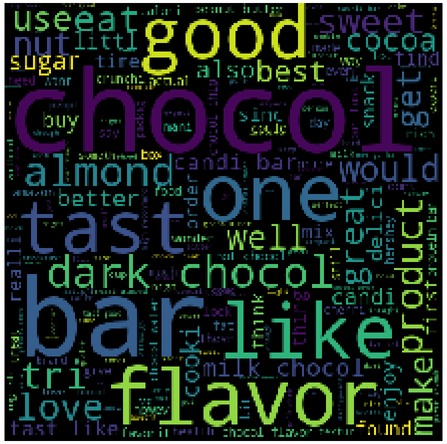


cluster = 13

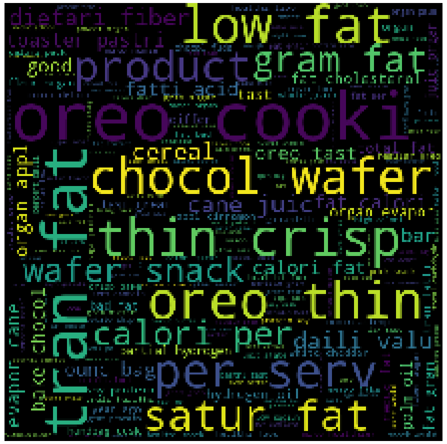


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

cluster = 14

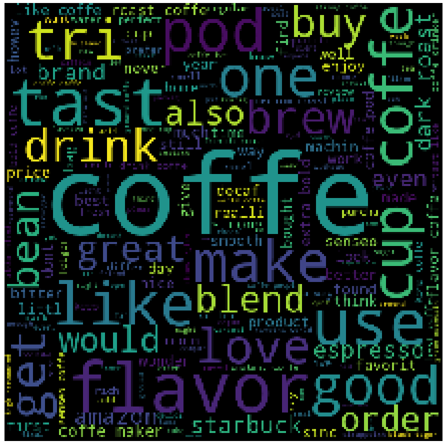


cluster = 15

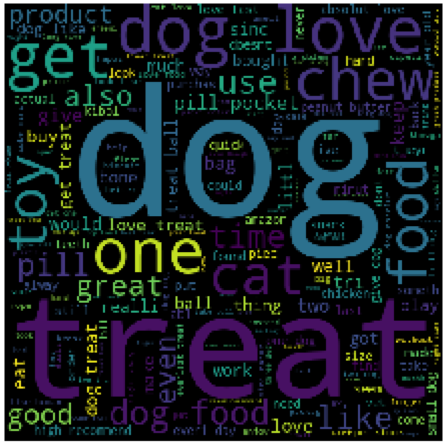


Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
cluster = 16
```

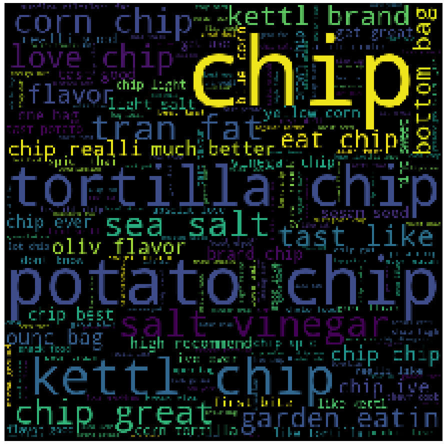



```
cluster = 17
```

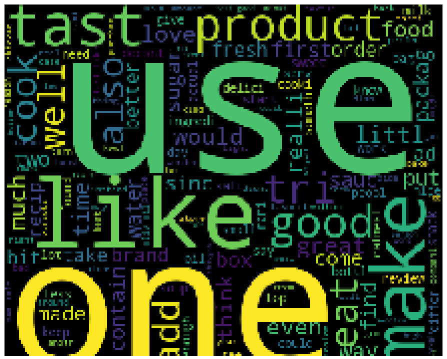



```
cluster = 18
```

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



cluster = 19



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

In [97]:
```
print(x)
```

```
+---------+------------------------------------------------------------------------
--------------------------------------------------------------------------
----------------------------------+
| CLUSTER |
                OBSERVATION
                                 |
+---------+------------------------------------------------------------------------
--------------------------------------------------------------------------
----------------------------------+
|    0    |                  Cluster - 0 talks about all the health re
lated things, here in word cloud we can see heart rate , metabolism, we
ight loss etc                    |
|    1    |                                  Cluster 1 talks about all
the food item names like cocunut oil , olive oil , sugar , scoop etc
                                |
|    2    |                                            This cluster ma
inly contains natural ingredients like quinoa and saponin..
                                 |
|    3    |                                              This cluster
contains all the reviews which contain the word food....
                                 |
|    4    |                                             This cluster c
ontains all the reviews which tell the product experience
                                 |
|    5    |                                            This cluster co
ntains all negative points like   dont .. kill.. trap etc
                                 |
|    6    |                                           LIKE , GOOD , LO
VE , FLAVOUR... ALL This type of reviews are in this cluster
                                 |
|    7    |                                         This contain type
and taste of the tea.. like green tea, black tea, grey , etc
```

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

```
                                   |
|    8    |                      Even this contain about the taste of
tea , as well as the colour.. it contains some words like , highly reco
mmond etc                           |
|    9    |                                                           I
Think cluster 9 and cluster 6 are more similar
                                    |
|    10   |                                        This contains state
of the product as it is liquid and soda type cool  , fizz etc
                                   |
|    11   |
               Duplicate of 8
                                    |
|    12   |
              Duplicate of 7,8
                                    |
|    13   |                                                  It contai
ns all the reviews which contain a word cat.. mostly
                                    |
|    14   |                                                    ALL the
food items reviews are here, like almond , chocol
                                   |
|    15   | here we have all the reviews stating the fat present in the
product like low fat, satur fat etc and also about items like oreo and
also reviews of size  thin etc  |
|    16   |                                                    This is
like , highly positive review love,like drink etc
                                   |
|    17   |                                               Even this is
something about taste of the product , like chewing etc
                                    |
|    18   |                                                This contai
ns all the reviews ,which contain the word chip in it
                                    |
|    19   |                                                          T
his is again same like highly positive review
                                    |
+---------+-------------------------------------------------------------
```

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

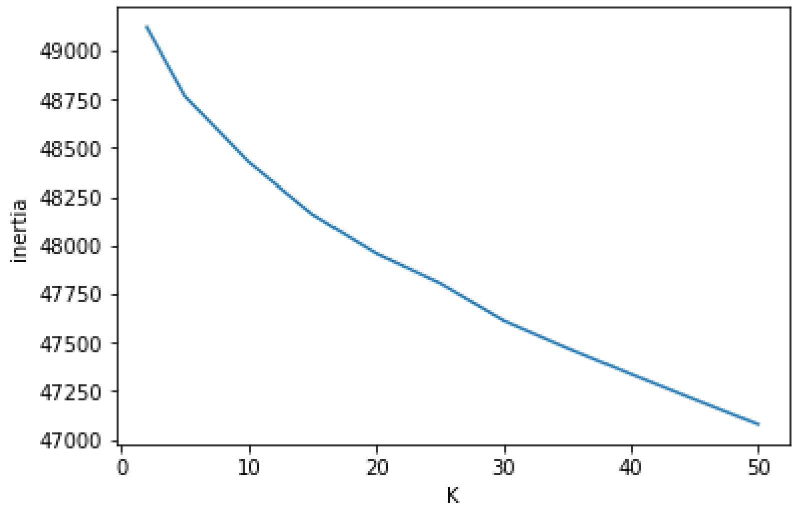
--------------------------------+

In [12]:
```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2),min_df=10)
tf_idf_vect1 = TfidfVectorizer(ngram_range=(1,2))
final_tf_idf = tf_idf_vect.fit(sampledata['CleanedText'].values)
final_tf_idf1 = tf_idf_vect1.fit(sampledata['CleanedText'].values)
```

In [13]:
```
X_tf_idf = final_tf_idf.transform(sampledata['CleanedText'].values)
```

In [44]:
```
from sklearn.cluster import KMeans
K = [2,5,10,15,20,25,30,35,40,45,50]
inertia = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0).fit(X_tf_idf)
    inertia.append(kmeans.inertia_)
```

In [45]:
```
plt.plot(K,inertia)
plt.xlabel('K')
plt.ylabel('inertia')
plt.show()
```



In [14]:

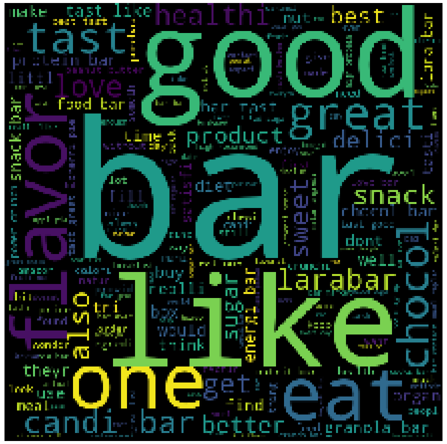
```
kmeans = KMeans(n_clusters=20, random_state=0).fit(X_tf_idf)
```

In [15]:

```
count=0
reviews=sampledata['CleanedText'].values
topn_class1 = sorted(zip(kmeans.labels_, reviews))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=500,height=500).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=500,
height=500).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = 0
```

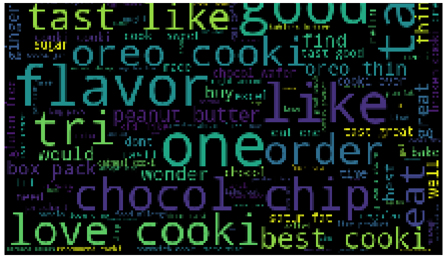
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



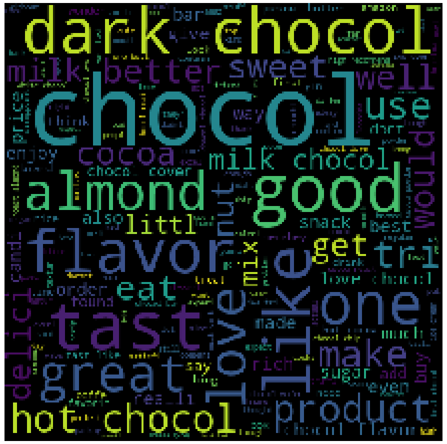
cluster = 1

cluster = 2

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



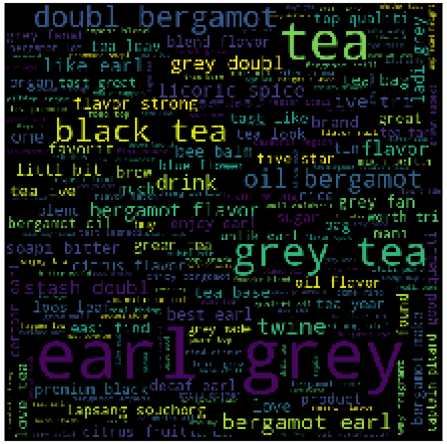
cluster = 3



cluster = 4

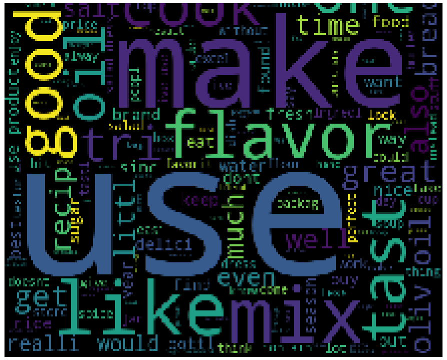
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

cluster = 5

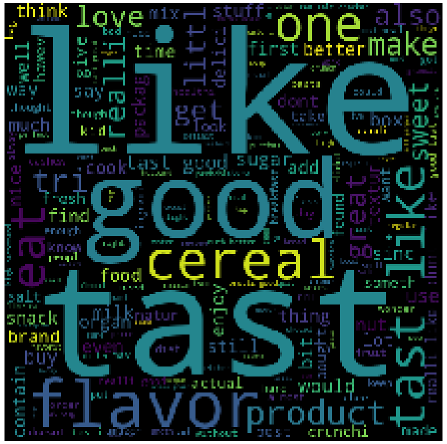


cluster = 6

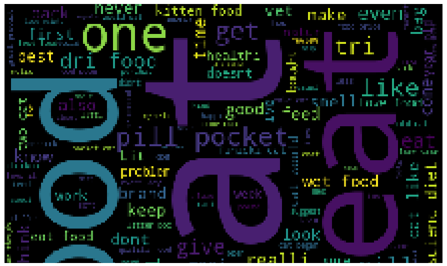
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



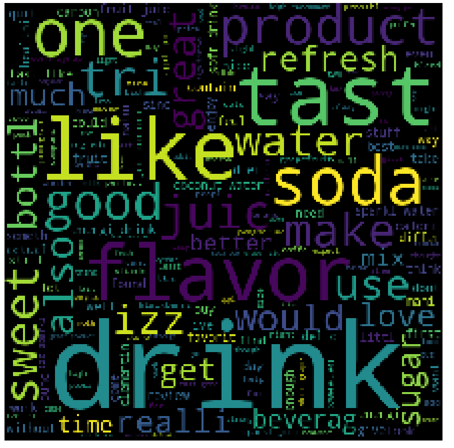

```
cluster = 7
```




```
cluster = 8
```



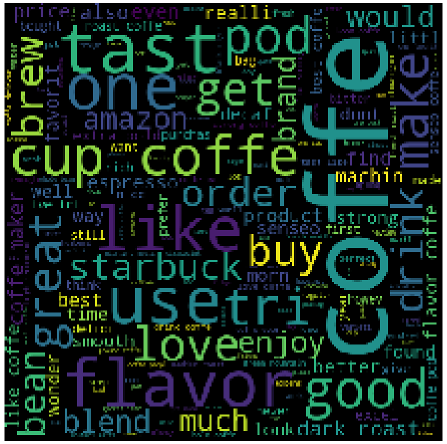
```
cluster = 9
```



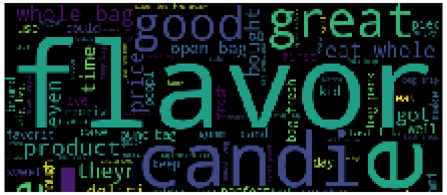

```
cluster = 10
```

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

cluster = 11

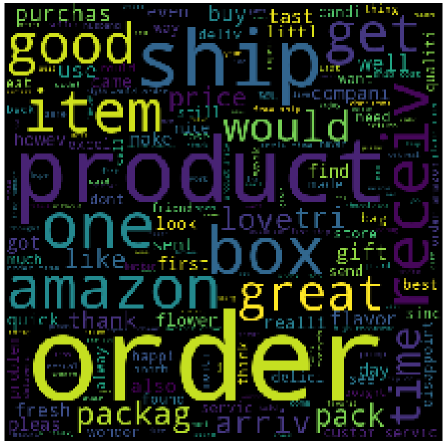


cluster = 12



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

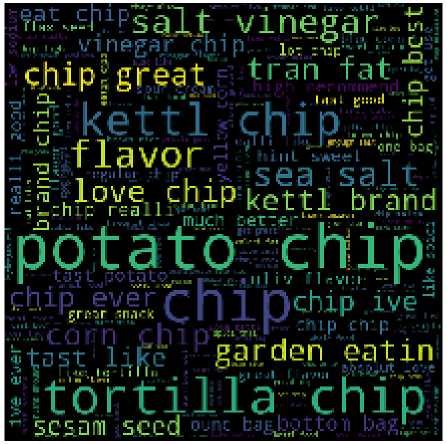

```
cluster = 13
```




```
cluster = 14
```

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

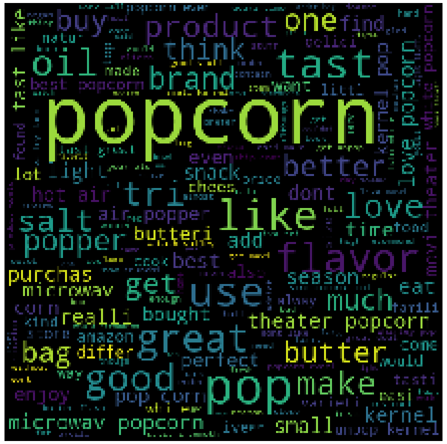
cluster = 15



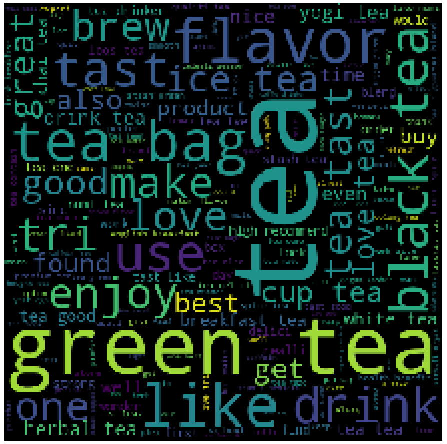
cluster = 16

cluster = 17

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



cluster = 18



cluster = 19

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

Cluster-0,1,8,12,14,16 is all about positive reviews, where we have words like good, like etc, cluster-2 is about all the food items like cooki , chocol , chip, oreo etc.. cluster-3 is about the reviews with type of chocol used like dark chocol,hot chocol , and also about items like almond etc.. I think, cluster-4 is all about that it is also found in local store like we have local groceri , food store etc, cluster - 5 is all about the reviews with type of tea like grey tea , black tea etc.. cluster -6,7 is about the taste of the product which are positive.. cluster 9 is about the taste as well as the way the product is.. liquid, drink , fizz etc.. cluster 10 , i think it has all the differentreviews as it is a mixture of all..cluster-11 are the reviews which compare it with coffee , cluster-13 is about shipping of amazon and the way it is shipped in box..cluster-15 is about, the reviews which contain the word chip in it ... like potato chip, love chip , tortilla chip..cluster-17 is about combination with the product -- popcorn , pop etc.. cluster 18 and 19 are about all the positive reviews and flavour ....

In [16]:
```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
i=0
list_of_sent=[]
for sent in sampledata['CleanedText'].values:
    list_of_sent.append(sent.split())
w2v_model=Word2Vec(list_of_sent,min_count=5,size=50, workers=4)
w2v_words = list(w2v_model.wv.vocab)
```

In [17]:
```
from tqdm import tqdm
```

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

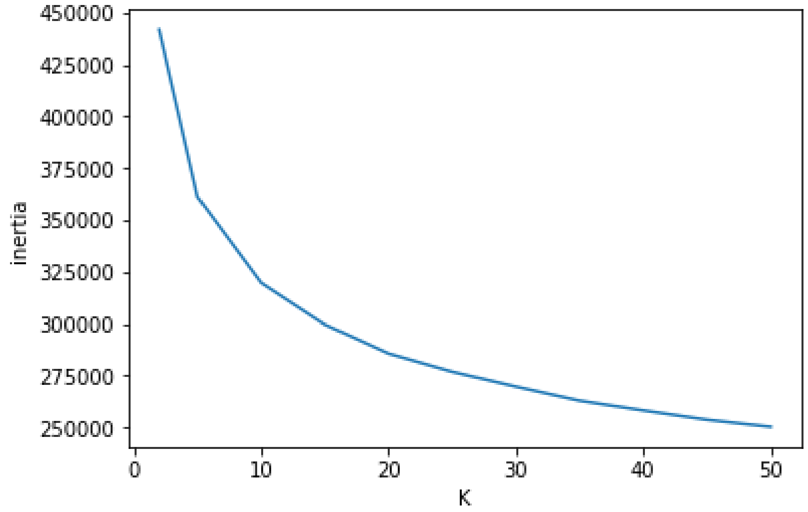
```
import os
sent_vectors = []; # the avg-w2v for each sentence/review is stored in
 this list
for sent in tqdm(list_of_sent): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/re
view
    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)
```

```
100%|██████████████████████████████████████████████████████████████████
█████████| 50000/50000 [01:03<00:00, 790.02it/s]
```

In [52]:
```
from sklearn.cluster import KMeans
K = [2,5,10,15,20,25,30,35,40,45,50]
inertia = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0).fit(sent_vectors)
    inertia.append(kmeans.inertia_)
```

In [53]:
```
plt.plot(K,inertia)
plt.xlabel('K')
plt.ylabel('inertia')
plt.show()
```

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



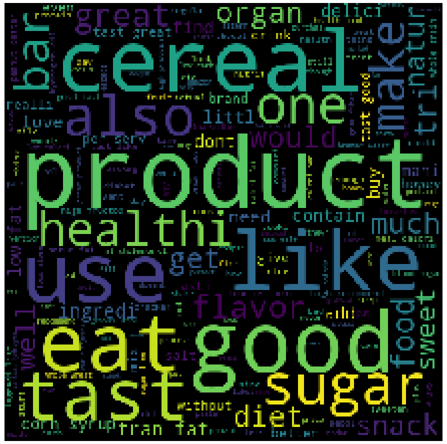
In [18]:
```
kmeans1 = KMeans(n_clusters=20, random_state=0).fit(sent_vectors)
```

In [19]:
```
count=0
reviews=sampledata['CleanedText'].values
topn_class1 = sorted(zip(kmeans1.labels_, reviews))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=500,height=500).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
```

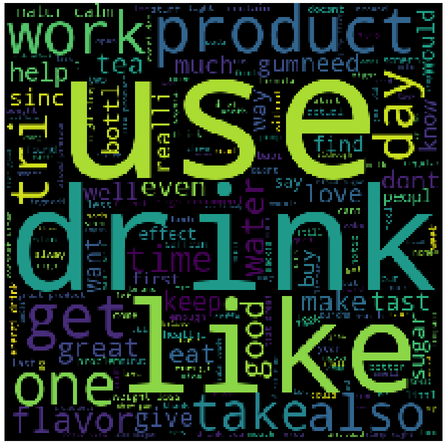
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=500,
height=500).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = 0
```




```
cluster = 1
```



cluster = 2

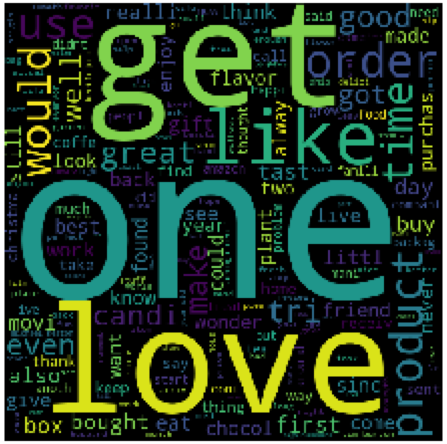
cluster = 3

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

cluster = 4

cluster = 5

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

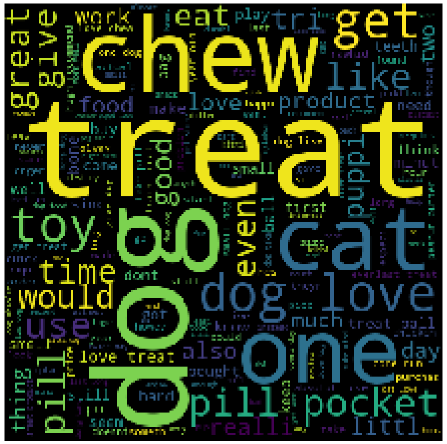


cluster = 6

cluster = 7

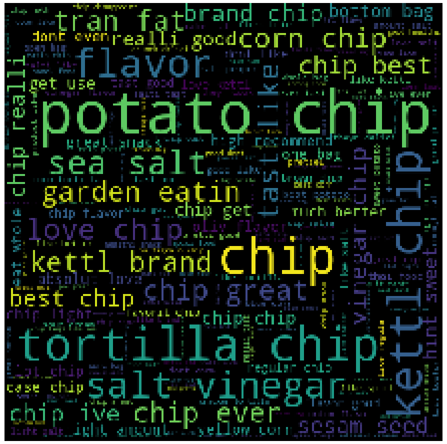
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

cluster = 8



cluster = 9

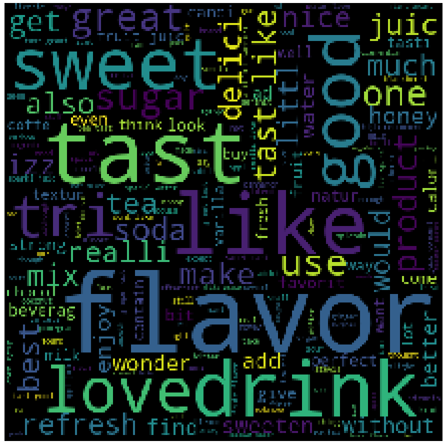
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



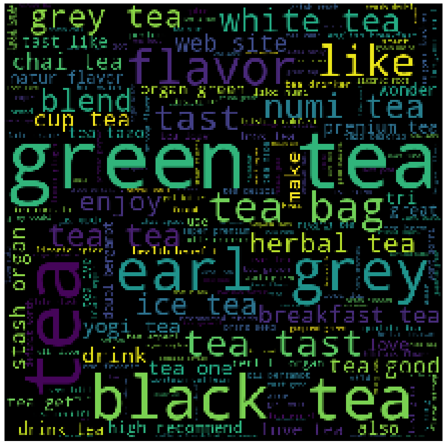
cluster = 10

cluster = 11

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

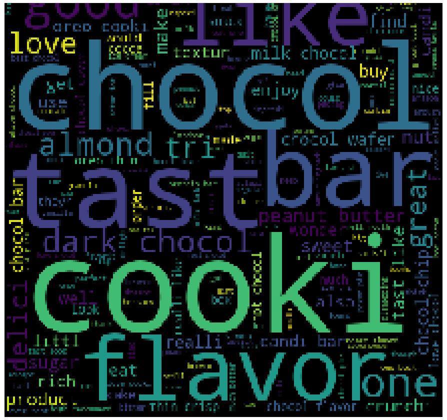


cluster = 12



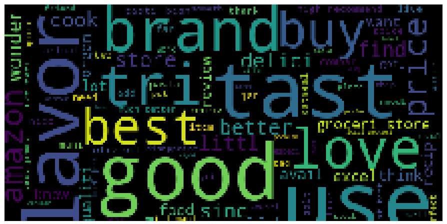
cluster = 13

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

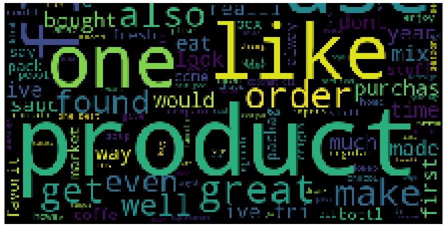


cluster = 14

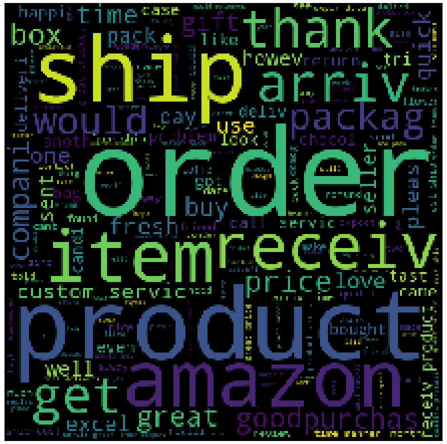
cluster = 15



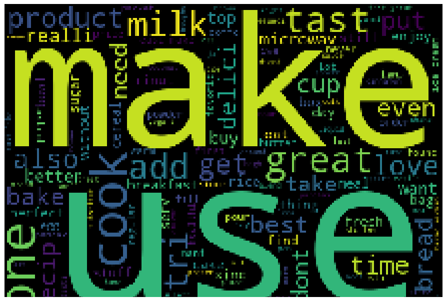
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



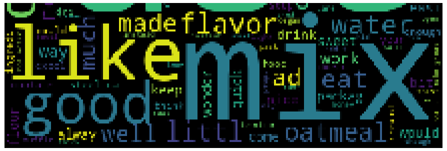
cluster = 16



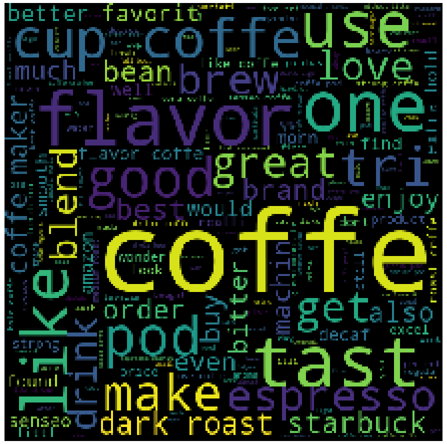
cluster = 17



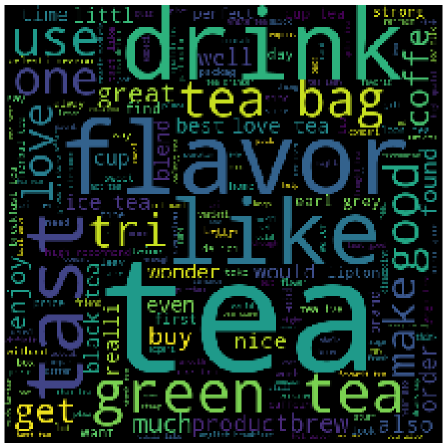
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



cluster = 18



cluster = 19



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

Cluster -0,2,5,6,10,11,12,14,15 It contains some of the food items reviews and and some positive .... Cluster-1,19 contains its state and also positive reviews..... Cluster - 3 contains about packing ..reviews about packing in bag box etc cluster - 4 is about price and the thing it says is that it is also available in local stores .. cluster - 7,8 contains all the reviews which specify animals and with word food and how to eat it.. cluster - 9 contains the word chip in it.. potato chip , kettl chip etc , cluster -11 contains reviews with type of tea, green tea , grey tea etc , cluster -13 contains all the items which are in it like chocol, cooki , almond etc cluster -16 is about delivery of the item... cluster 17 is about how to prepare like mix milk cook etc cluster-18 is comparing with coffee

In [20]:
```
from tqdm import tqdm
import os
# TF-IDF weighted Word2Vec

tfidf_feat = tf_idf_vect.get_feature_names()
dictionary = dict(zip(tf_idf_vect1.get_feature_names(), list(tf_idf_vec
t1.idf_)))# 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_sent): # 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:
            vec = w2v_model.wv[word]
            # obtain the tf_idfidf of a word in a sentence/review
            tf_idf = dictionary[word]*(sent.count(word)/len(sent))
            sent_vec += (vec * tf_idf)
            weight_sum += tf_idf
    if weight_sum != 0:
        sent_vec /= weight_sum
    tfidf_sent_vectors.append(sent_vec)
    row += 1
```

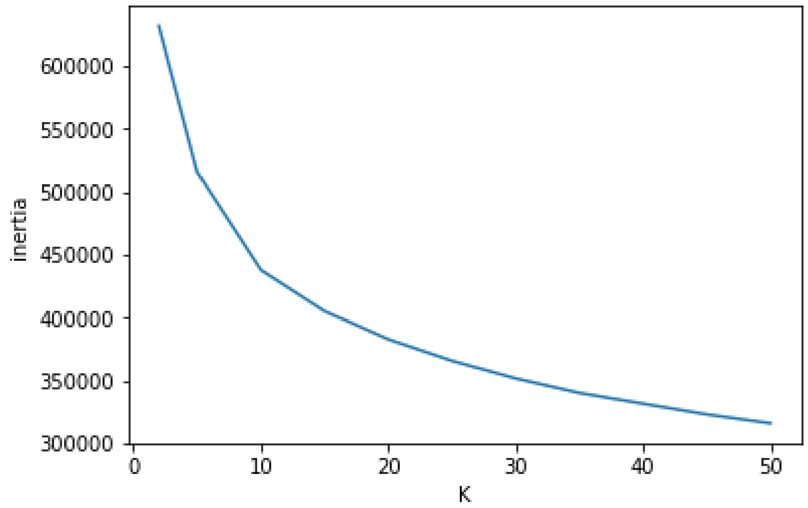
```
100%|██████████████████████████████████████████████████
```

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

```
██████████| 50000/50000 [01:27<00:00, 571.57it/s]
```

In [58]:
```python
from sklearn.cluster import KMeans
K = [2,5,10,15,20,25,30,35,40,45,50]
inertia = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0).fit(tfidf_sent_vector
s)
    inertia.append(kmeans.inertia_)
```

In [59]:
```python
plt.plot(K,inertia)
plt.xlabel('K')
plt.ylabel('inertia')
plt.show()
```



In [21]:
```python
kmeans2 = KMeans(n_clusters=20, random_state=0).fit(tfidf_sent_vectors)
```

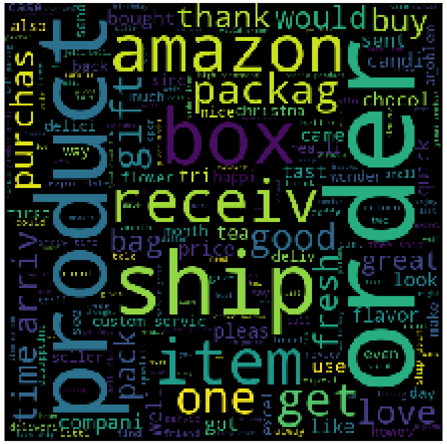
In [22]:
```python
count=0
reviews=sampledata['CleanedText'].values
topn_class1 = sorted(zip(kmeans2.labels_, reviews))
feature =[]
```

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

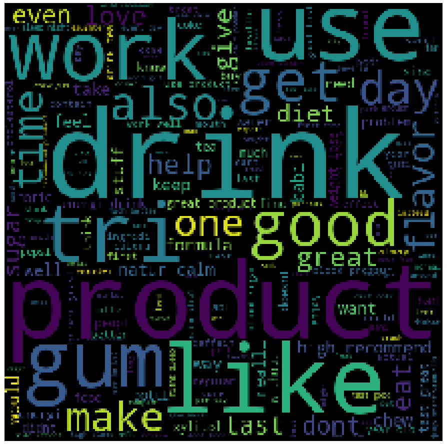
```
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=500,height=500).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=500,
height=500).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = 0
```

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




```
cluster = 1
```

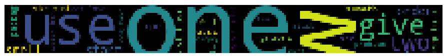



```
cluster = 2
```

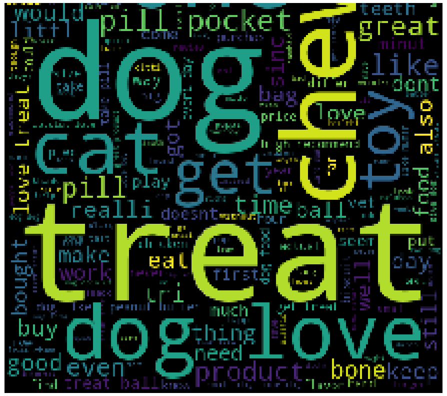
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

cluster = 3

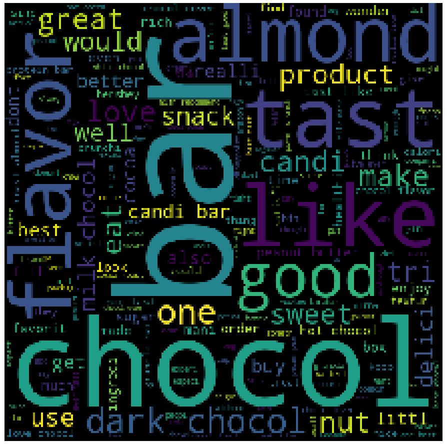
cluster = 4



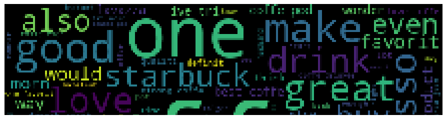
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



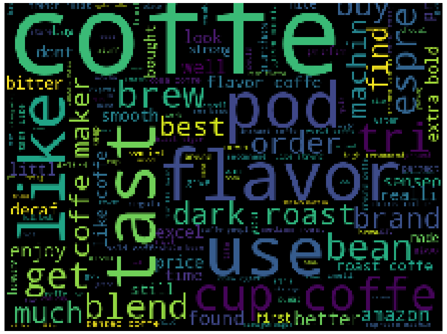
cluster = 5



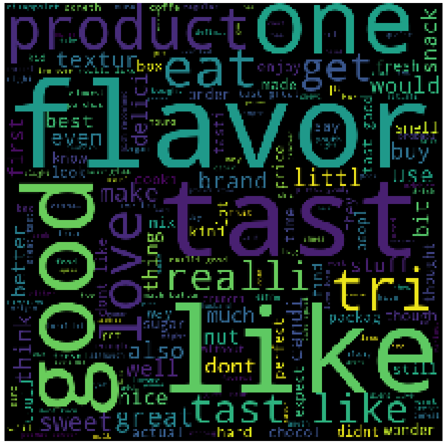
cluster = 6



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

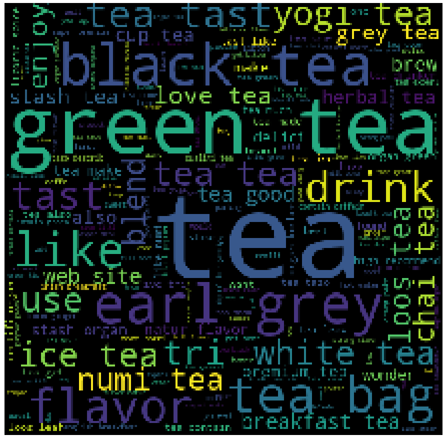


cluster = 7

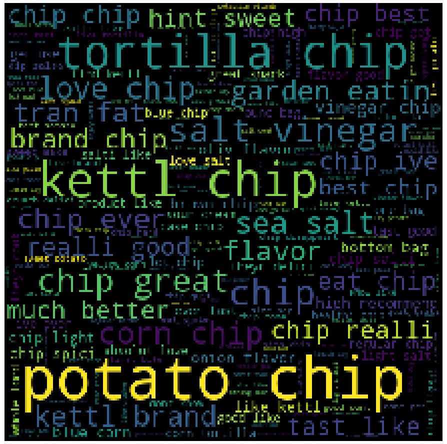


cluster = 8

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



cluster = 9

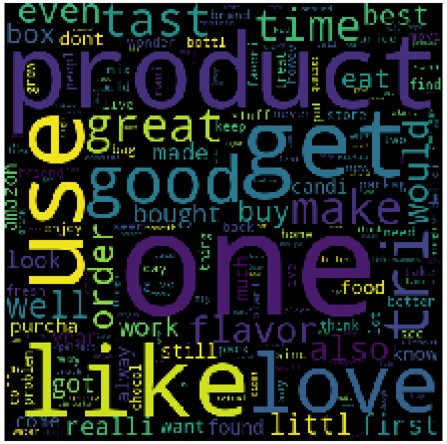


cluster = 10

cluster = 11

cluster = 12

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

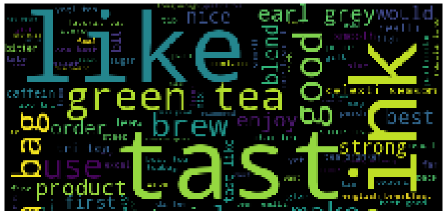


cluster = 13

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

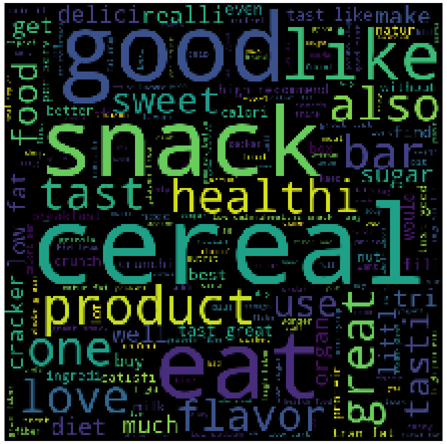
```
cluster = 14
```


```
cluster = 15
```



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

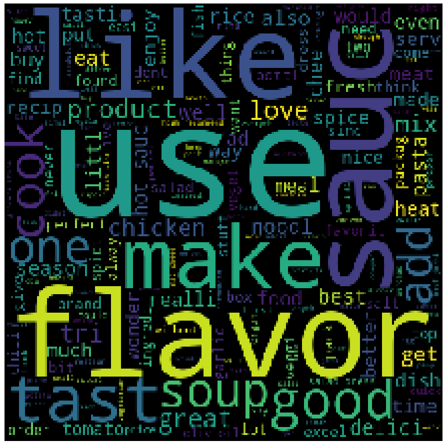
cluster = 16



cluster = 17

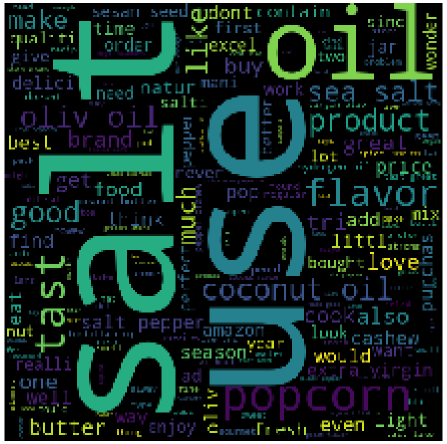
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

cluster = 18



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

```
cluster = 19
```



cluster-0 is about Product, shipping of product.. boxbag etc..cluster -1,19 is about the state of product , gum, drink etc... cluster-2 is all about the positive reviews about the product the packaging in bag is lovable etc..cluster-3 is about that it is also found in groceri store and something about price and amazon... cluster-4 is positive as well as it is also something about animals cat , dog etc..cluster-5 is about food items like almond , chocol taste etc..cluster-6 is positive about the product and comparing with the coffee.. cluster-7,11,17,18 is about positive about product taste .. flavour, tast , like good etc cluster- 8,15 is about type of tea ... green tea, grey tea and ice tea...cluster-9 is all about chip word kettl chip, brand chip etc... Cluster- 10 is about cooki and size of the cooki is thin crisp....cluster 13 is all about food, pet food dog , cat etc.. cluster 14 is procedure to cook that food..cluster-16 is positive as well as the way it can be taken ass snacks etc

Agglomerative Clustering on AVG-W2V and TF-IDF W2V

In [23]:
```python
sampledata1 = sorted_data.head(5000)
```

In [24]:
```python
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
```

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

```
i=0
list_of_sent1=[]
for sent in sampledata1['CleanedText'].values:
    list_of_sent1.append(sent.split())
w2v_model=Word2Vec(list_of_sent1,min_count=5,size=50, workers=4)
w2v_words = list(w2v_model.wv.vocab)
```

In [25]:

```
from tqdm import tqdm
import os
sent_vectors1 = []; # the avg-w2v for each sentence/review is stored in
 this list
for sent in tqdm(list_of_sent1): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    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_vectors1.append(sent_vec)
```

```
100%|██████████████████████████████████████████████████
██████████| 5000/5000 [00:06<00:00, 797.11it/s]
```

In [44]:

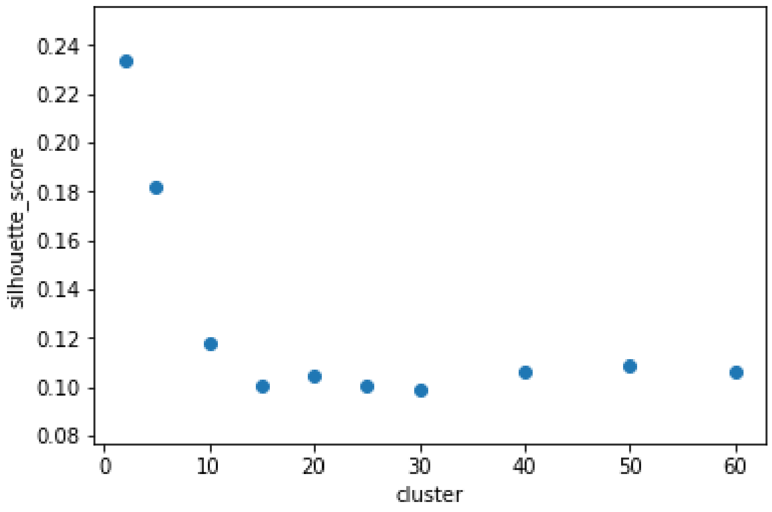
```
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
cluster = [2,5,10,15,20,25,30,40,50,60]
met =[]
for k in cluster:
    clu = AgglomerativeClustering(n_clusters=k).fit(sent_vectors1)
    met.append(silhouette_score(sent_vectors1,clu.labels_))
```

In [46]:

```
plt.scatter(cluster,met)
plt.xlabel('cluster')
```

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

```
plt.ylabel('silhouette_score')
plt.show()
```



In [26]:
```
from sklearn.cluster import AgglomerativeClustering
clu = AgglomerativeClustering(n_clusters=2).fit(sent_vectors1)
```

In [27]:
```
clu.fit_predict(sent_vectors1)
```

Out[27]:
```
array([1, 1, 1, ..., 0, 1, 1], dtype=int64)
```

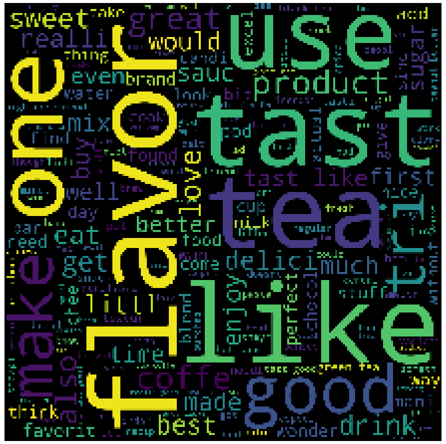
In [28]:
```
def againcleaning(X):
    comment_words=' '
    for words in X:
        comment_words = comment_words + words + ' '
    return comment_words
count=0
reviews=sampledata1['CleanedText'].values
topn_class1 = sorted(zip(clu.labels_, reviews))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
```

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

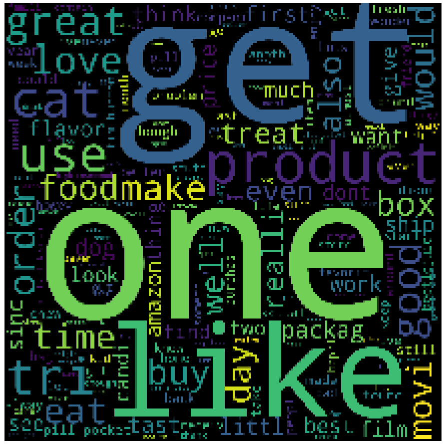
```
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=1200,height=1200).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=1200
,height=1200).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = 0
```

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




```
cluster = 1
```



Cluster-1 is focussing on all the aspects like flavour taste and alll and why does the reviewers like..tea, taste, flavour ,use Cluster 2 it mainly focuses on the reviews where people love it and some reviews are like get,try etc

In [29]:
```
tf_idf_vect2 = TfidfVectorizer(ngram_range=(1,2))
final_tf_idf2 = tf_idf_vect2.fit(sampledata1['CleanedText'].values)
```

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

In [30]:
```python
from tqdm import tqdm
import os
# TF-IDF weighted Word2Vec

tfidf_feat = tf_idf_vect2.get_feature_names()
dictionary = dict(zip(tf_idf_vect2.get_feature_names(), list(tf_idf_vec
t2.idf_)))# tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_sent_vectors1 = []; # the tfidf-w2v for each sentence/review is s
tored in this list
row=0;
for sent in tqdm(list_of_sent1): # 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:
            vec = w2v_model.wv[word]
            # obtain the tf_idfidf of a word in a sentence/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_vectors1.append(sent_vec)
    row += 1
```

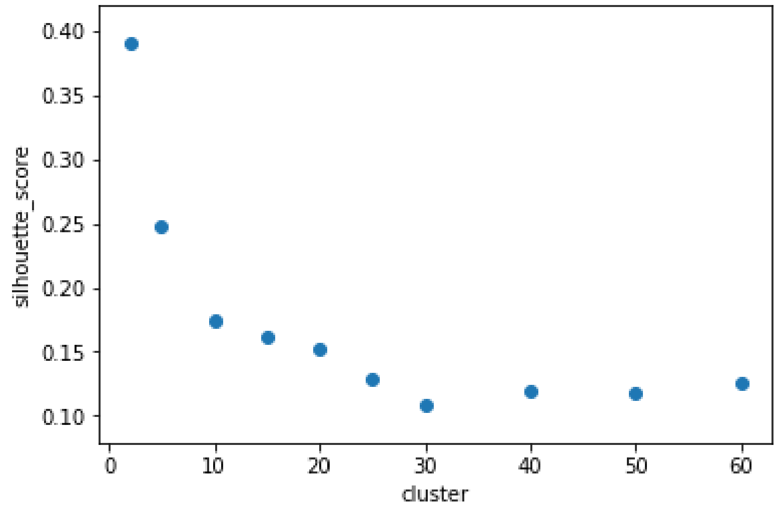
```
100%|██████████████████████████████████████████████████████████████████
███████████| 5000/5000 [00:08<00:00, 611.70it/s]
```

In [27]:
```python
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
cluster = [2,5,10,15,20,25,30,40,50,60]
met =[]
for k in cluster:
    clu = AgglomerativeClustering(n_clusters=k).fit(tfidf_sent_vectors1
```

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

```
)
    met.append(silhouette_score(tfidf_sent_vectors1,clu.labels_))
```

In [28]:
```
plt.scatter(cluster,met)
plt.xlabel('cluster')
plt.ylabel('silhouette_score')
plt.show()
```



In [31]:
```
clu = AgglomerativeClustering(n_clusters=2).fit(tfidf_sent_vectors1)
clu.fit_predict(tfidf_sent_vectors1)
```

Out[31]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

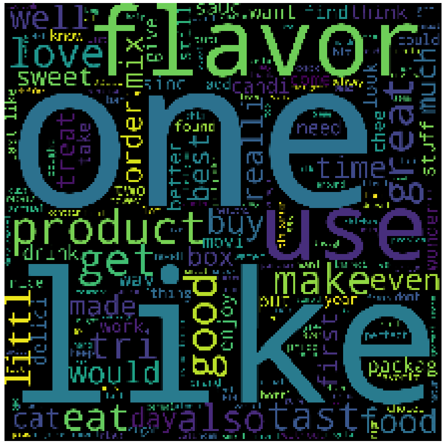
In [32]:
```
count=0
reviews=sampledata1['CleanedText'].values
topn_class1 = sorted(zip(clu.labels_, reviews))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
```

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

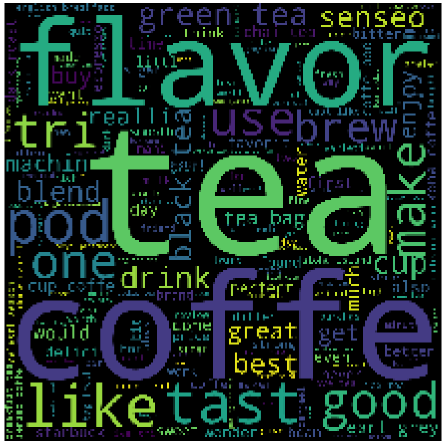
```
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=1200,height=1200).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=1200
,height=1200).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = 0
```

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




```
cluster = 1
```



cluster-0 is all about liking , get it etc while cluster - 1 contains tea , word that is specifications of the product , shape etc and comparing it with coffee etcand specifying that it has good shape

DBSCAN on Avg W2V and TF-IDF W2V

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

I am using the following to find best eps... 3) sensitivity analysis Basically we want to chose a radius that is able to cluster more truly regular points (points that are similar to other points), while at the same time detect out more noise (outlier points). We can draw a percentage of regular points (points belong to a cluster) VS. epsilon analysis, where we set different epsilon values as the x-axis, and their corresponding percentage of regular points as the y axis, and hopefully we can spot a segment where the percentage of regular points value is more sensitive to the epsilon value, and we choose the upper bound epsilon value as our optimal parameter.

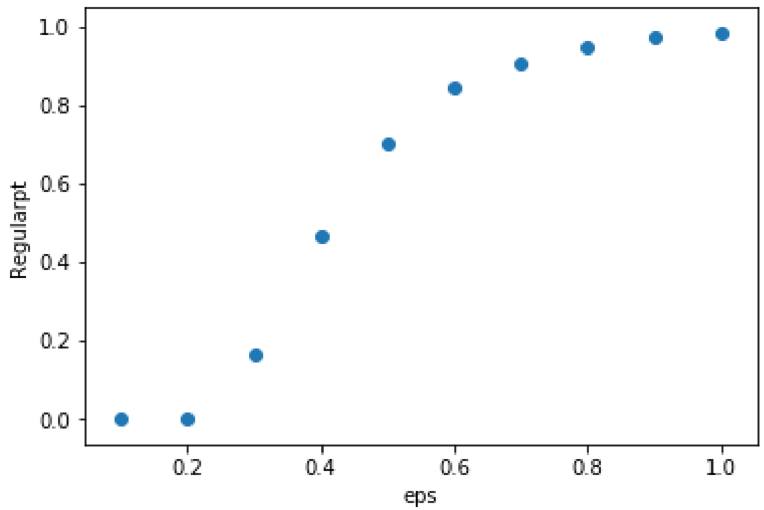
In [68]:
```
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import silhouette_score
eps = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
Regularpt = []
for k in eps:
    clu = DBSCAN(eps=k,min_samples=100).fit(sent_vectors1)
    sampl = len(clu.core_sample_indices_)
    avgsampl = float(sampl/5000)
    Regularpt.append(avgsampl)
    #met = NearestNeighbors(n_neighbors=300,radius=k).fit(sent_vectors
1)
    #kthdistance.append(knn(met,sent_vectors1))
```

In [44]:
```
len(sampl)
```

Out[44]: 5000

In [69]:
```
plt.scatter(eps,Regularpt)
plt.xlabel('eps')
plt.ylabel('Regularpt')
plt.show()
```

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



In [85]:
```
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
clu = DBSCAN(eps=0.6,min_samples=100).fit(sent_vectors1)
```

In [86]:
```
clu.labels_
```

Out[86]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

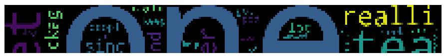
In [87]:
```
count=-1
reviews=sampledata1['CleanedText'].values
topn_class1 = sorted(zip(clu.labels_, reviews))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=1200,height=1200).generate(a)
```

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

```
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=1200
,height=1200).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = -1
```


```
cluster = 0
```



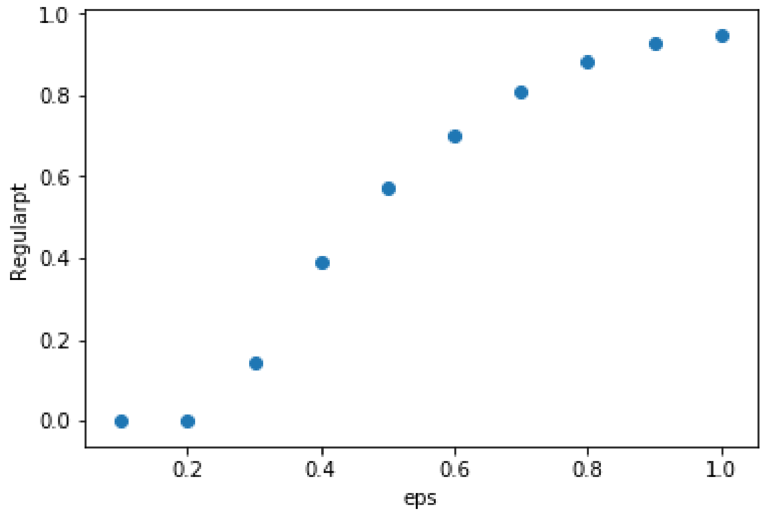
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

It is differentiating generally noise from imp reviews I am using the following to find best eps... 3) sensitivity analysis Basically we want to chose a radius that is able to cluster more truly regular points (points that are similar to other points), while at the same time detect out more noise (outlier points). We can draw a percentage of regular points (points belong to a cluster) VS. epsilon analysis, where we set different epsilon values as the x-axis, and their corresponding percentage of regular points as the y axis, and hopefully we can spot a segment where the percentage of regular points value is more sensitive to the epsilon value, and we choose the upper bound epsilon value as our optimal parameter.

In [88]:
```
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import silhouette_score
eps = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
Regularpt = []
for k in eps:
    clu = DBSCAN(eps=k,min_samples=100).fit(tfidf_sent_vectors1)
    sampl = len(clu.core_sample_indices_)
    avgsampl = float(sampl/5000)
    Regularpt.append(avgsampl)
```

In [89]:
```
plt.scatter(eps,Regularpt)
plt.xlabel('eps')
plt.ylabel('Regularpt')
plt.show()
```

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



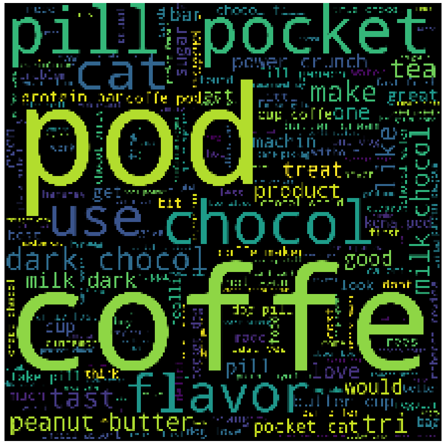
In [90]:
```
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
clu = DBSCAN(eps=0.8,min_samples=100).fit(tfidf_sent_vectors1)
```

In [91]:
```
count=-1
reviews=sampledata1['CleanedText'].values
topn_class1 = sorted(zip(clu.labels_, reviews))
feature =[]
for coef,feat in topn_class1:
        if coef == count:
            feature.append(feat)
        else:
            a=againcleaning(feature)
            print(" cluster =", count)
            from wordcloud import WordCloud, STOPWORDS
            import matplotlib.pyplot as plt
            word_cloud=WordCloud(background_color='black',stopwords=sto
p,width=1200,height=1200).generate(a)
            plt.imshow(word_cloud)
            plt.axis("off")
            plt.show()
            count = count + 1
```

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

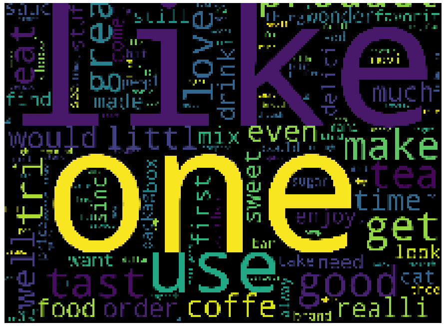
```
            feature =[]
            feature.append(feat)
#for label = 19
a=againcleaning(feature)
print(" cluster =", count)
#print(a, " " , count)
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
word_cloud=WordCloud(background_color='black',stopwords=stop,width=1200
,height=1200).generate(a)
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

```
cluster = -1
```




```
cluster = 0
```

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



Differentiating noise from imp reviews cluster -1 is noise

Conclusion

In Kmeans i have made them in to 20 clusters, where i can really find the differences between the clusters clearly... as it was said... In Agglomerative clustering where we used silhouette_score and got no. of clusters when it is 2 we have hign silhouette_score so we used that.. For eps in dbscan , I took it as per sensitivity analysis it will easily seperate noise from imp reviws.....

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