

NOTE:

1. K-means clustering perform by using 2-lakh samples.

1. DBSCAN and Agglomerative clustering perform by using 20k samples.

Import necessary libraries

In [1]:

```
1 import warnings  
2 warnings.filterwarnings('ignore')
```

In [28]:

```
1 from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN  
2 from sklearn.neighbors import NearestNeighbors  
3 from sklearn.preprocessing import StandardScaler  
4 from sklearn.metrics import *  
5 import pickle  
6 from tqdm import tqdm  
7 import seaborn as sns  
8 import numpy as np  
9 import matplotlib.pyplot as plt  
10 import pandas as pd  
11 from scipy.sparse import *  
12 from wordcloud import WordCloud  
13 from operator import itemgetter  
14 from os import path  
15 from PIL import Image  
16 import os  
17 import requests  
18  
19 %matplotlib inline
```

Load preprocessed data

```
In [49]: 1 #Functions to save objects for later use and retrieve it
2 def savetofile(obj,filename):
3     pickle.dump(obj,open(filename+".pkl","wb"))
4 def openfromfile(filename):
5     temp = pickle.load(open(filename+".pkl","rb"))
6     return temp
7
8 #DATA FOR K-MEANS
9 #X=openfromfile('X')
10
11 #count_vect =openfromfile('count_vect')
12 #X_bigram = openfromfile('X_bigram')
13
14 #tf_idf_vect =openfromfile('tf_idf_vect')
15 #X_tfidf =openfromfile('X_tfidf')
16
17 #avg_sent_vectors=openfromfile('avg_sent_vectors')
18 #tfidf_sent_vectors=openfromfile('tfidf_sent_vectors')
19
20
21 #DATA FOR AGGLOMERATIVE AND DBSCAN
22 X_agg=openfromfile('X_agg')
23
24 avg_sent_vectors_agg=openfromfile('avg_sent_vectors_agg')
25 tfidf_sent_vectors_agg=openfromfile('tfidf_sent_vectors_agg')
26
27
```

```
In [50]: 1 print('No. of points used for Agglomerative and DBSCAN: ',len(avg_sent_vectors_agg))
2 #print('No. of points used for K-means: ',len(avg_sent_vectors))
```

No. of points used for Agglomerative and DBSCAN: 19354

```
In [51]: 1 X_agg.shape
```

Out[51]: (19354,)

Standardizing data

```
In [52]: 1 def std_data(data,mean):  
2     scaler=StandardScaler(with_mean=mean)  
3     std_data=scaler.fit_transform(data)  
4     return std_data
```

Clustering:

Function for K-Means hyperparameter tunnning using elbow method :

```
In [53]: 1 def clustering_model(data,params,vect):  
2     '''hyperparam tunnning and draw elbow curve'''  
3     inertia_value=[]  
4     for k in tqdm(params['clusters']):  
5         model=KMeans(n_clusters=k,init='k-means++',n_jobs=-1)  
6         model.fit(data)  
7         inertia_value.append(model.inertia_)  
8     plt.figure(1,figsize=(10,6))  
9     plt.plot(params['clusters'],inertia_value)  
10    plt.title('ELBOW METHOD %s(OPTIMAL HYPERPARAM)'%vect)  
11    plt.xlabel('K(No. of clusters): Hyperparam')  
12    plt.ylabel('Inertia(Sum of intra cluster distance)')  
13    plt.grid(True)  
14    plt.show()  
15
```

Function for K-distance-plot for eps tunnning:

```
In [54]: 1 def k_dist_plot(data):
2     neigh=data.shape[1] * 2 #200
3     nbrs=NearestNeighbors(n_neighbors=neigh).fit(data)
4     dist,ind= nbrs.kneighbors(data)
5     distance_dec_order=sorted(dist[:,neigh-1],reverse=False)
6     plt.figure(1,figsize=(8,5))
7     plt.grid(True)
8     plt.title('ELBOW CURVE (EPS TUNNING)')
9     plt.xlabel('No. Of Points')
10    plt.ylabel('Distance to %dth NN '%neigh)
11    plt.plot(list(range(1,data.shape[0] + 1)), distance_dec_order)
12    #distanceDec=sorted(dist[:ns-1],reverse=True)
```

```
In [55]: 1 data.shape
```

```
Out[55]: (19354, 100)
```

Function for labeling reviews and visualizing no. of reviews per cluster :

```
In [56]: 1 def review_labelling(data,all_Reviews,params,vect,algo):
2     if algo=='kmeans':
3         model=KMeans(n_clusters=params,precompute_distances=True,init='k-means++',n_jobs=-1)
4     elif algo=='agglomerative':
5         model=AgglomerativeClustering(n_clusters=params)
6     elif algo=='dbSCAN':
7         model=DBSCAN(eps=params,min_samples=200 , n_jobs=-1)
8
9     model.fit(data)
10    labels=model.labels_
11    cluster_label=np.c_[ all_Reviews, labels ]
12    cluster_label_df=pd.DataFrame(cluster_label,columns=['review','label'])
13    print(cluster_label_df['label'].value_counts())
14    list_samples=[]
15    g=cluster_label_df.groupby(['label'])
16    for key in g.groups:
17        #AS DBSCAN GIVE -1 LABEL TO NOISY POINT
18        if key!= -1:
19            list_samples.append(len(g.groups[key]))
20    #BAR CHART
21    if len(list_samples)!=0:
22        plt.figure(1,figsize=(10,6))
23        sns.set(rc={'figure.figsize':(11.7,8.27)})
24        plt.title("NO. of Reviews per cluster")
25        # ADD BARS
26        plt.bar(range(len(list_samples)), list_samples) #plt.bar(range(params), list_samples)
27        # ADD LABELS
28        plt.xticks(range(len(list_samples)))
29        plt.xlabel('No. of Cluster')
30        plt.ylabel('No. of Reviews')
31        plt.show()
32    else:
33        print('All points are declared as noisy points for eps=% .2f' %params)
34    return model,cluster_label_df
```

Function To draw wordcloud for Frequent words belongs to each cluster :

```
In [57]: 1 def wordcloud_each_cluster(optimal_cluster,cluster_label_df):
2     corpus_cluster={}
3     #PROPERTIES OF TITLE
4     title_font = {'size': '20', 'color': 'black'}
5     if not os.path.isfile('mask-cloud.png'):
6         url='http://www.shapecollage.com/shapes/mask-cloud.png'
7         r=requests.get(url)
8         with open('mask-cloud.png', 'wb') as f:
9             f.write(r.content)
10    #SHAPE WORDCLOUD ACCORDING TO MASKABLE IMAGE
11    mask_image_path = path.dirname(__file__) if "__file__" in locals() else os.getcwd()
12    mask = np.array(Image.open(path.join(mask_image_path, "mask-cloud.png")))
13
14    for i in range(optimal_cluster):
15        #INITIALIZE WORDCLOUD OBJECT
16        wordcloud = WordCloud(max_font_size=120,\n17                                colormap='winter',\n18                                max_words=100,\n19                                collocations=False,\n20                                relative_scaling=1.0,\n21                                mask=mask,\n22                                background_color='white',\n23                                contour_width=2,\n24                                contour_color='royalblue')
15        #PICK REVIEWS BELONGS TO EACH CLUSTER
16        cluster_reviews=cluster_label_df.loc[cluster_label_df['label']==i,['review']]
17        #MERGE ALL REVIEWS BELONGS TO EACH CLUSTER INTO A SINGLE STRING
18        corpus_cluster[i]= ' '.join(list(cluster_reviews['review']))
19        #CREATE CORPUS OF WORDS FOR EACH CLUSTER
20        word_token=wordcloud.process_text(corpus_cluster[i])
21        #GENERATE WORDCLOUD ACCORDING TO WORD FREQUENCY
22        wordcloud.generate_from_frequencies(word_token)
23        plt.figure(1,figsize=(14,13))
24        plt.title("FREQUENT WORDS BELONGS TO CLUSTER-%d"%i, **title_font )
25        plt.imshow(wordcloud, interpolation="bilinear")
26        plt.axis("off")
27        plt.show()
```

Initialization of common objects required for all vectorizer

In [58]:

```
1 #VECTORIZER USED
2 vect=['BOW', 'TFIDF', 'AVG-W2V', 'TFIDF-W2V']
3 #ALGORITHM USED
4 algo=['kmeans', 'agglomerative', 'dbSCAN']
5 #RANGE OF HYPERPARAM
6 params={'clusters':[z for z in range(1,10,1)]}
```

[1] K-Means Clustering

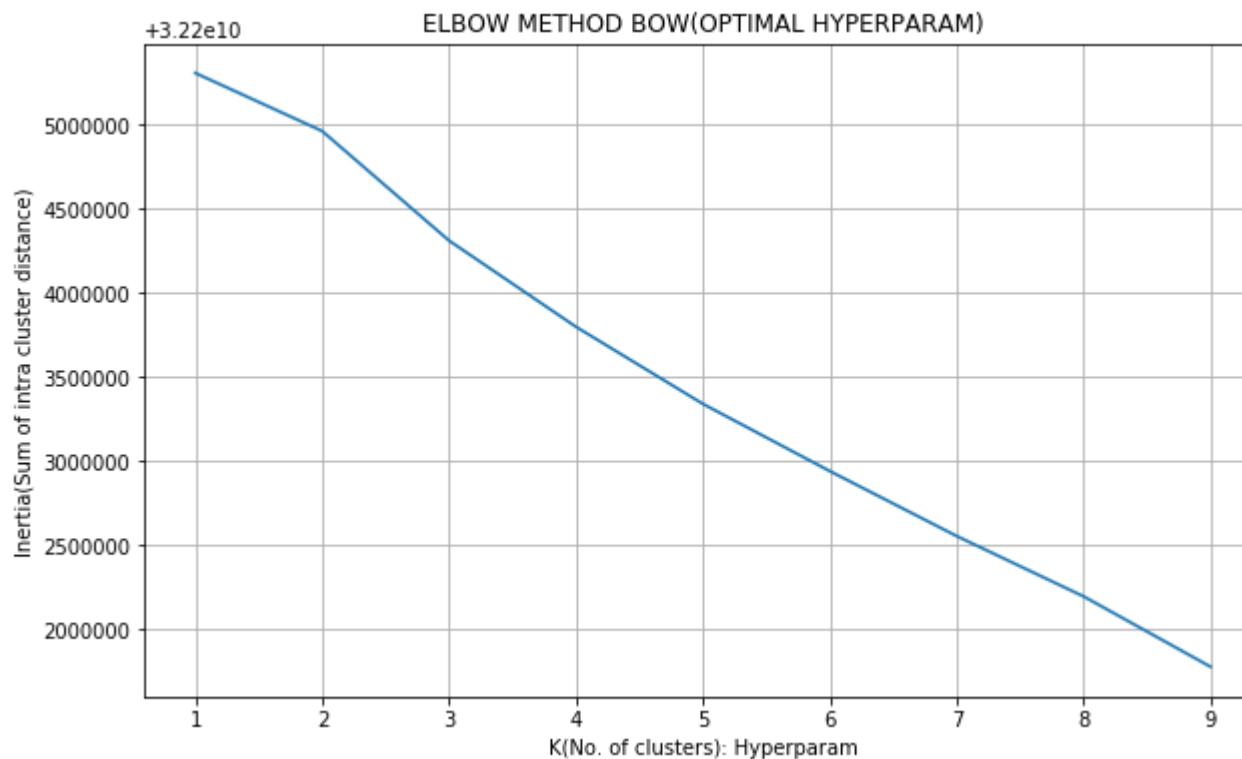
[1.1] Applying K-Means Clustering on BOW, SET 1

[1.1.1] Hyperparam Tunning using Elbow Method SET 1

In [11]:

```
1 data=std_data(X_bigram,mean=False)
2 %time clustering_model(data,params,vect[0])
```

100%|██████████| 9/9 [09:41<00:00, 65.37s/it]



CPU times: user 16min 14s, sys: 32min 33s, total: 48min 48s

Wall time: 9min 41s

Observation:

1. from the above plot we observe that after k=3, inertia reduces gradually so we choose k=3 as our knee point

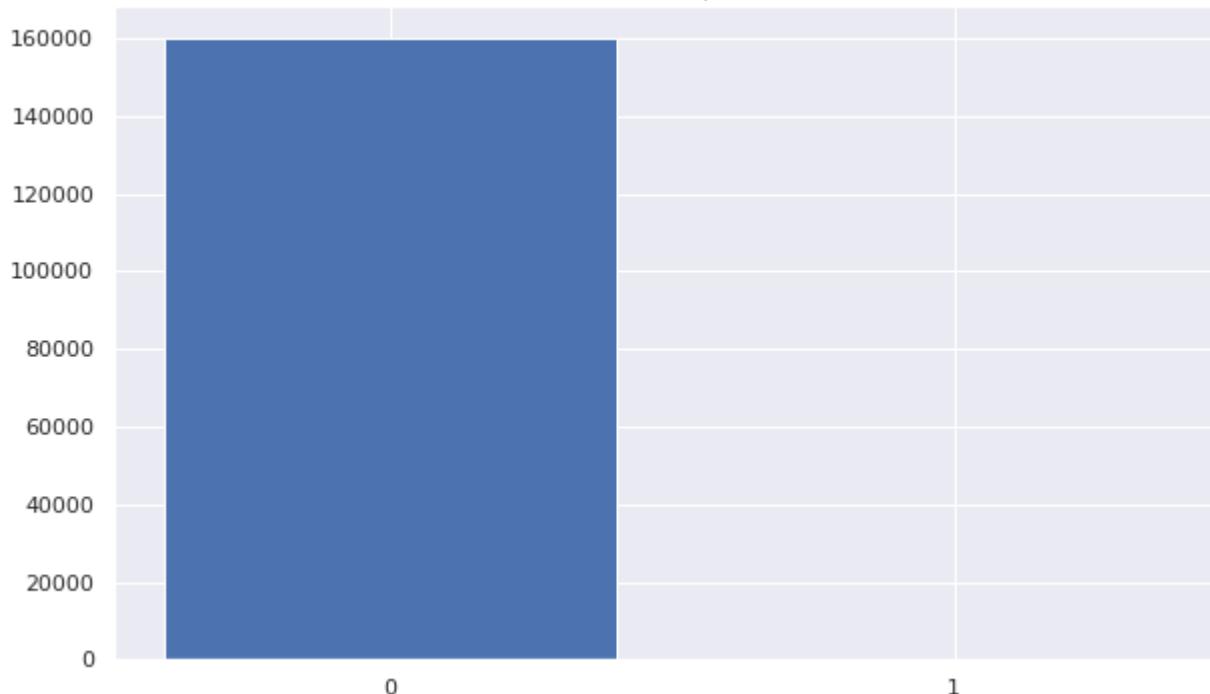
[1.1.2] Review labelling SET 1

In [17]:

```
1 optimal_cluster=2
2 model,cluster_label_df=review_labelling(data,X,optimal_cluster,vect[0],algo[0])
```

```
0    160148
1      28
Name: label, dtype: int64
```

NO. of Reviews per cluster



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

```
In [21]: 1 wordcloud_each_cluster(optimal_cluster,cluster_label_df)
```



- a. cluster-0 contains words which shows some positiveness(positive review)(ex. great, love, good, best etc)
- b. cluster-1 contains words related to chemical compounds(ex.calciu, sulfate, pyridoxine etc) and supplement(ex. vitamin, proteinate, grain, protein etc) products

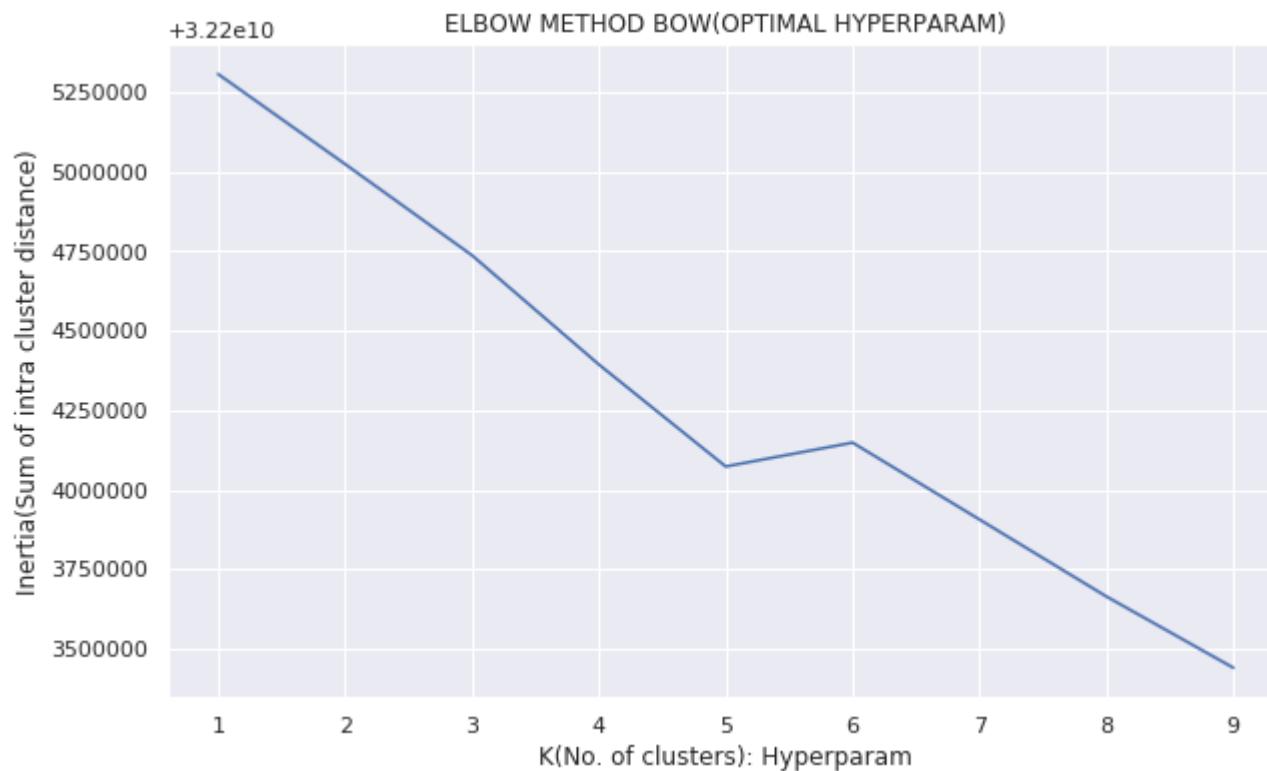
[1.2] Applying K-Means Clustering on TFIDF, **SET 2**

[1.2.1] Hyperparam Tunning **SET 2**

In [23]:

```
1 data=std_data(X_tfidf,mean=False)
2 %time clustering_model(data,params,vect[1])
```

100%|██████████| 9/9 [09:43<00:00, 65.22s/it]



CPU times: user 16min, sys: 30min 59s, total: 46min 59s

Wall time: 9min 43s

Observation:

- from the above plot we observe that after k=5, inertia reduces gradually so we choose k=5 as our knee point

[1.2.2] Review labelling SET 2

In [24]:

```
1 optimal_cluster=5
2 model,cluster_label_df=review_labelling(data,X,optimal_cluster,vect[0],algo[0])
2    160051
1      70
4      35
3      13
0       7
Name: label, dtype: int64
```

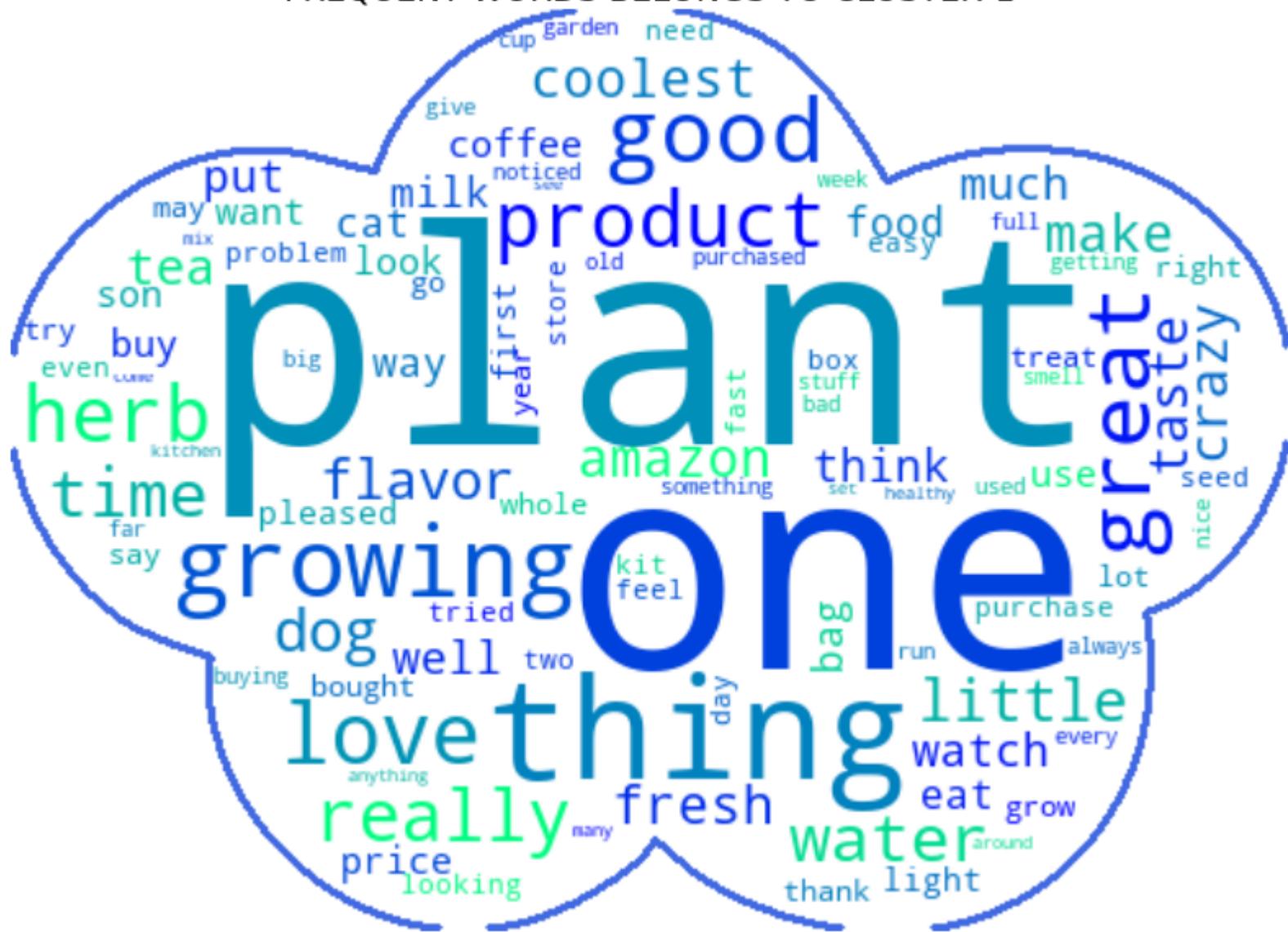


[1.2.3] Wordclouds of clusters obtained after applying k-means on TFIDF SET 2

```
In [30]: 1 wordcloud_each_cluster(optimal_cluster,cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1



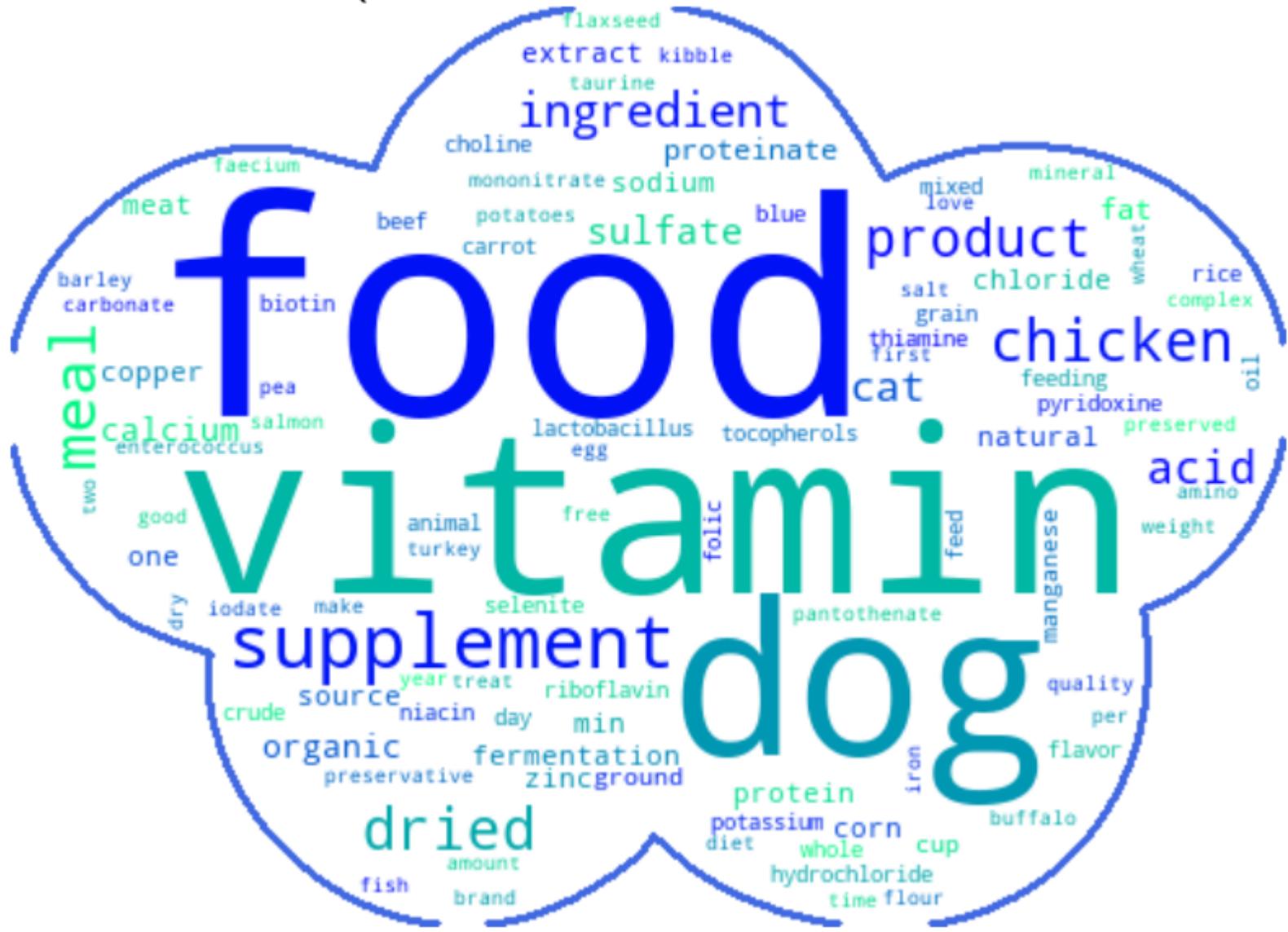
FREQUENT WORDS BELONGS TO CLUSTER-2



FREQUENT WORDS BELONGS TO CLUSTER-3



FREQUENT WORDS BELONGS TO CLUSTER-4

**Observation:**

1. From the above wordclouds we observe that:

- a. cluster-4 contains words related to chemical compounds(ex.caclium,sulfate,pyridoxine etc) and supplement(ex. vitamin, proteinate, grain, protein etc) products
- b. cluster-2 contains words which shows some positiveness(positive review)(ex. great, love, good, best etc)

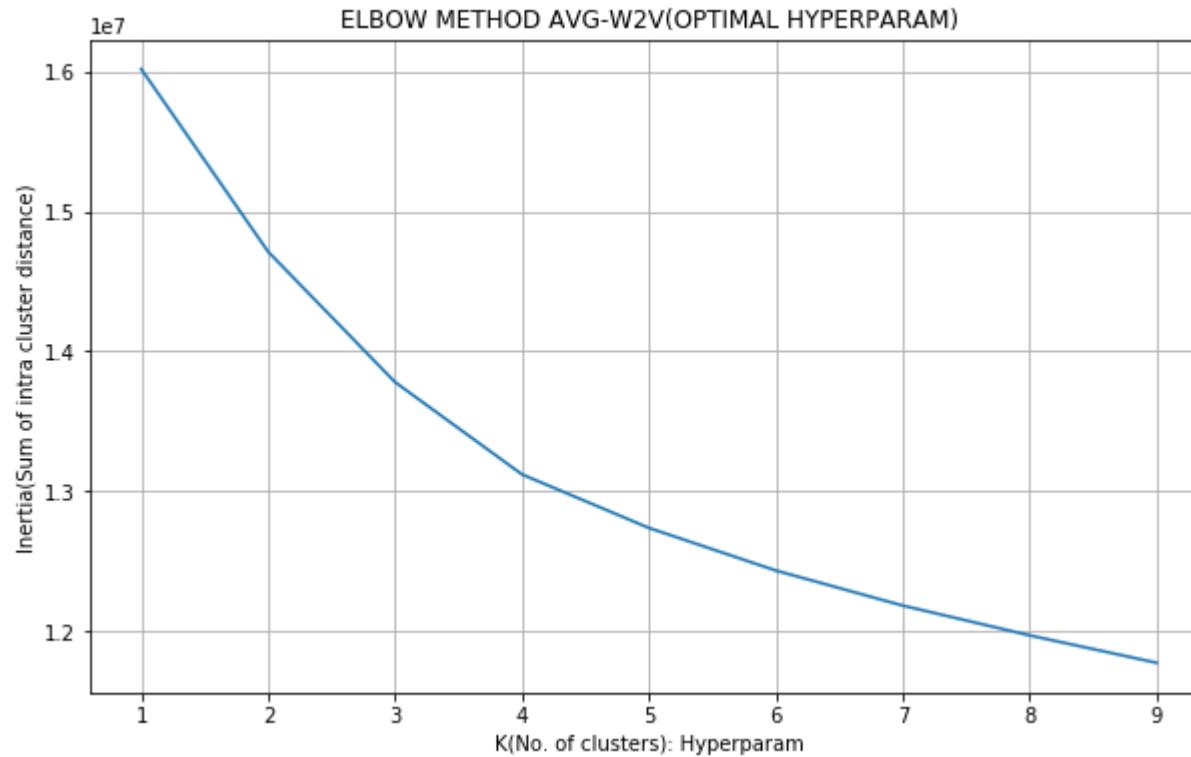
[1.3] Applying K-Means Clustering on AVG W2V, **SET 3**

[1.3.1] Hyperparam Tunning **SET 3**

In [10]:

```
1 params={'clusters':[z for z in range(1,10,1)]}
2 data=std_data(avg_sent_vectors,mean=True)
3 %time clustering_model(data,params,vec[2])
```

100%|██████████| 9/9 [15:09<00:00, 12
8.62s/it]



Wall time: 15min 10s

Observation:

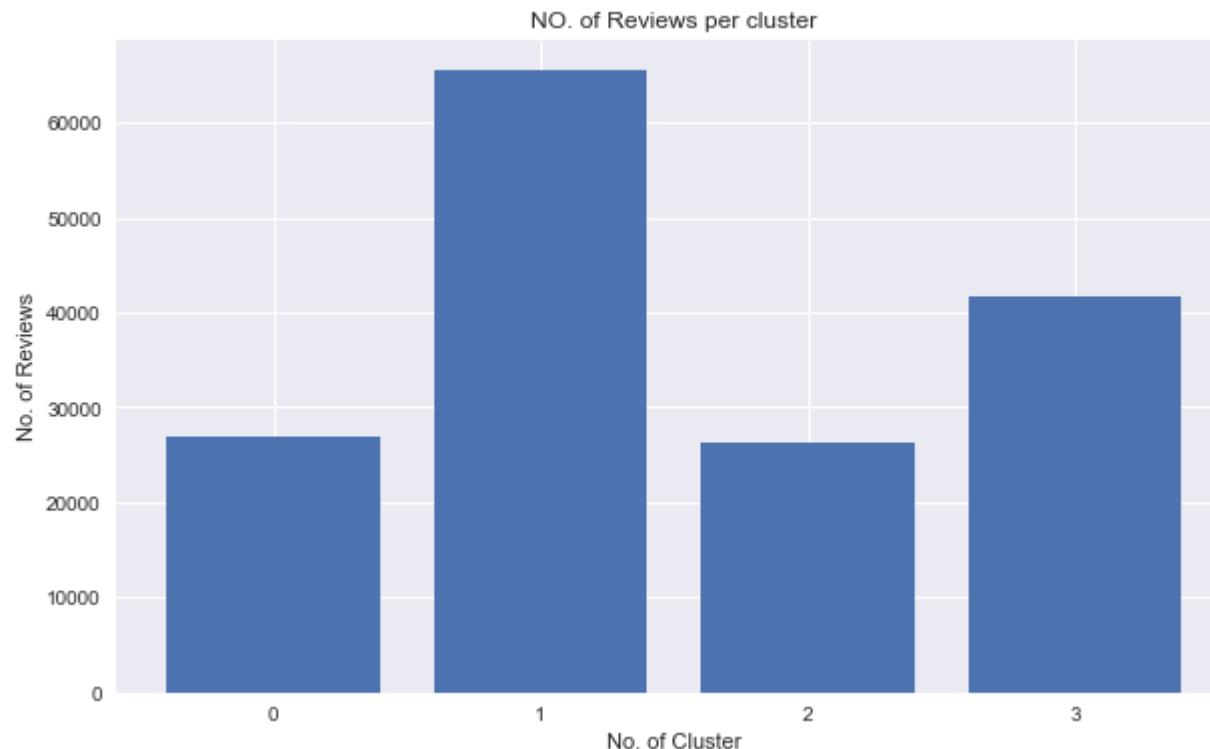
1. from the above plot we observe that after k=4, inertia reduces gradually so we choose k=4 as our knee point

[1.3.2] Review labelling SET 3

In [11]:

```
1 optimal_cluster=4
2 model,cluster_label_df=review_labelling(data,X,optimal_cluster,vect[2],algo[0])
```

```
1 65469
3 41676
0 26831
2 26200
Name: label, dtype: int64
```



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

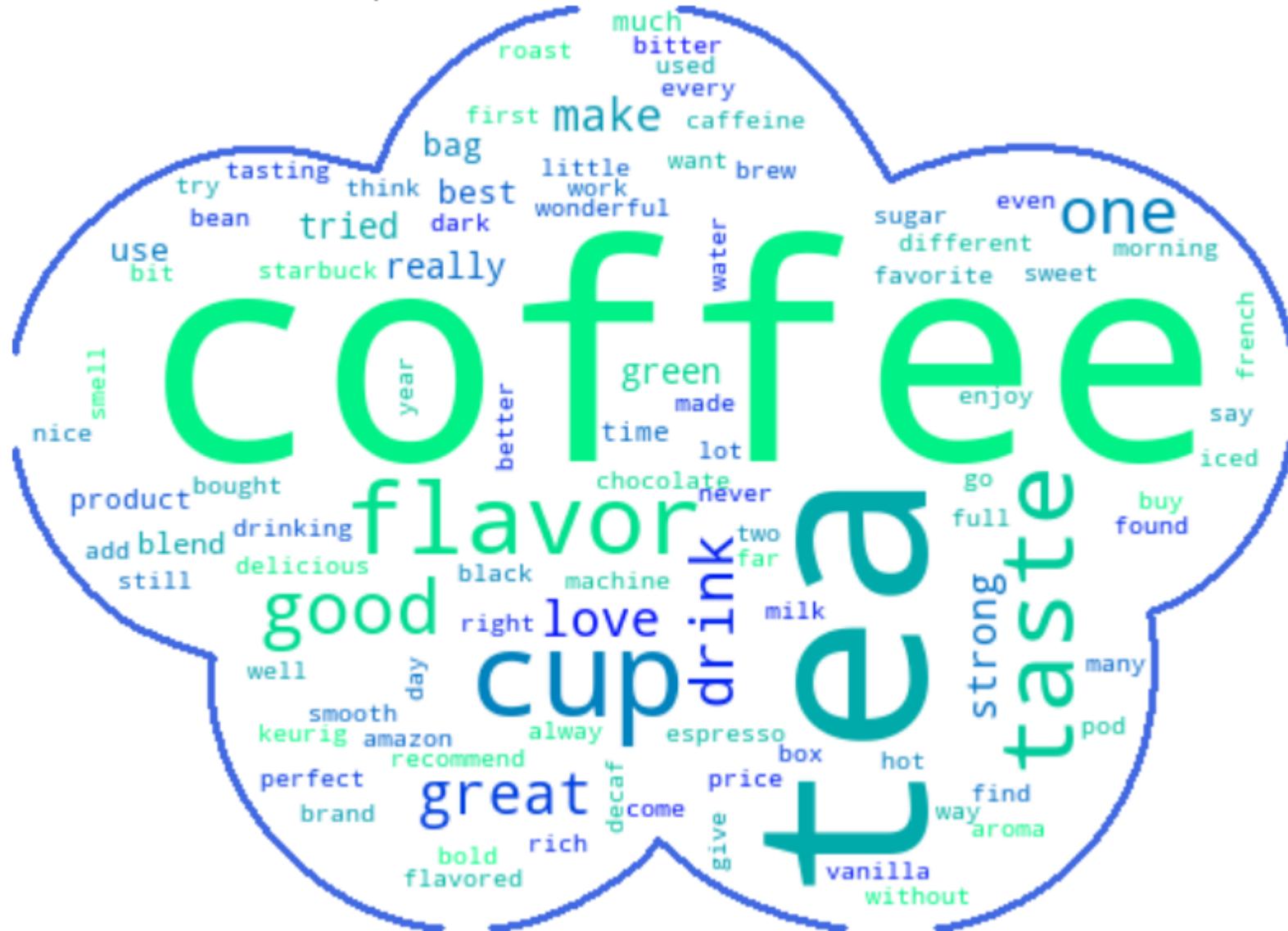
```
In [12]: 1 wordcloud_each_cluster(optimal_cluster,cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1



FREQUENT WORDS BELONGS TO CLUSTER-2



FREQUENT WORDS BELONGS TO CLUSTER-3

**Observation:**

1. From the above wordclouds we cant say anything about cluster because all the clusters are having mixup of words.

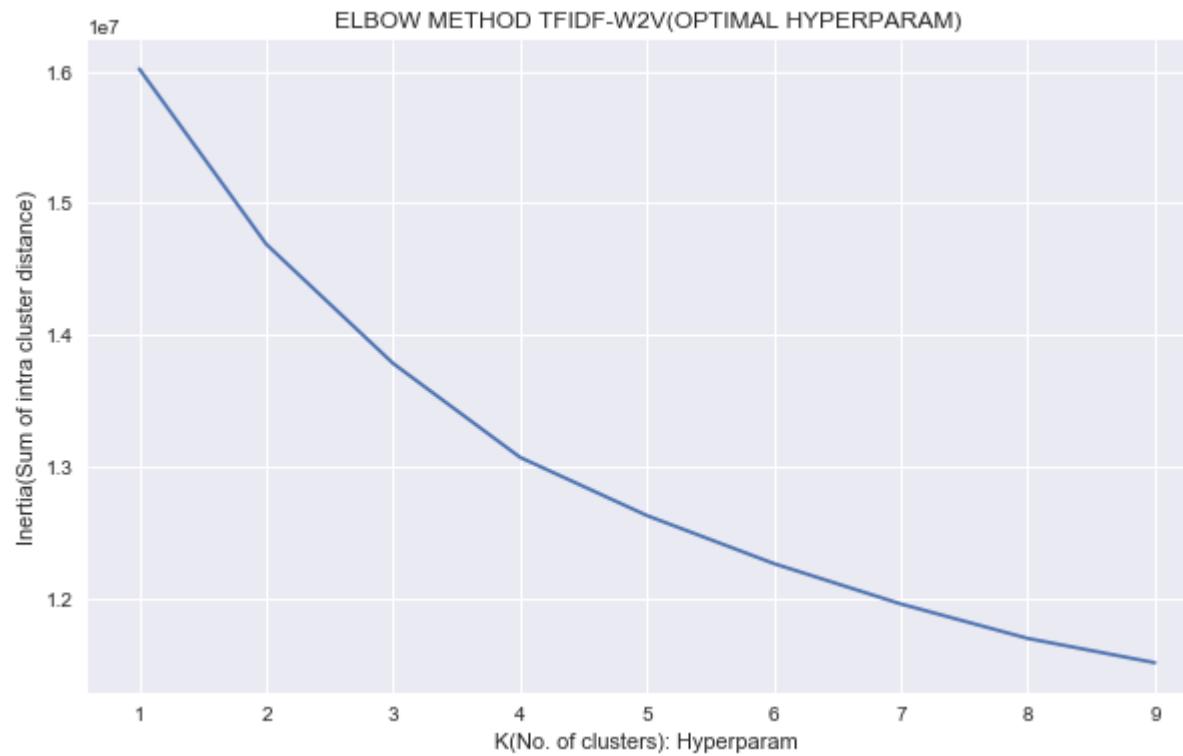
[1.4] Applying K-Means Clustering on TFIDF W2V, SET 4

[1.4.1] Hyperparam Tunning SET 4

In [13]:

```
1 data=std_data(tfidf_sent_vectors,mean=True)
2 %time clustering_model(data,params,vect[3])
```

100% | 9/9 [17:43<00:00, 14
5.34s/it]



Wall time: 17min 44s

Observation:

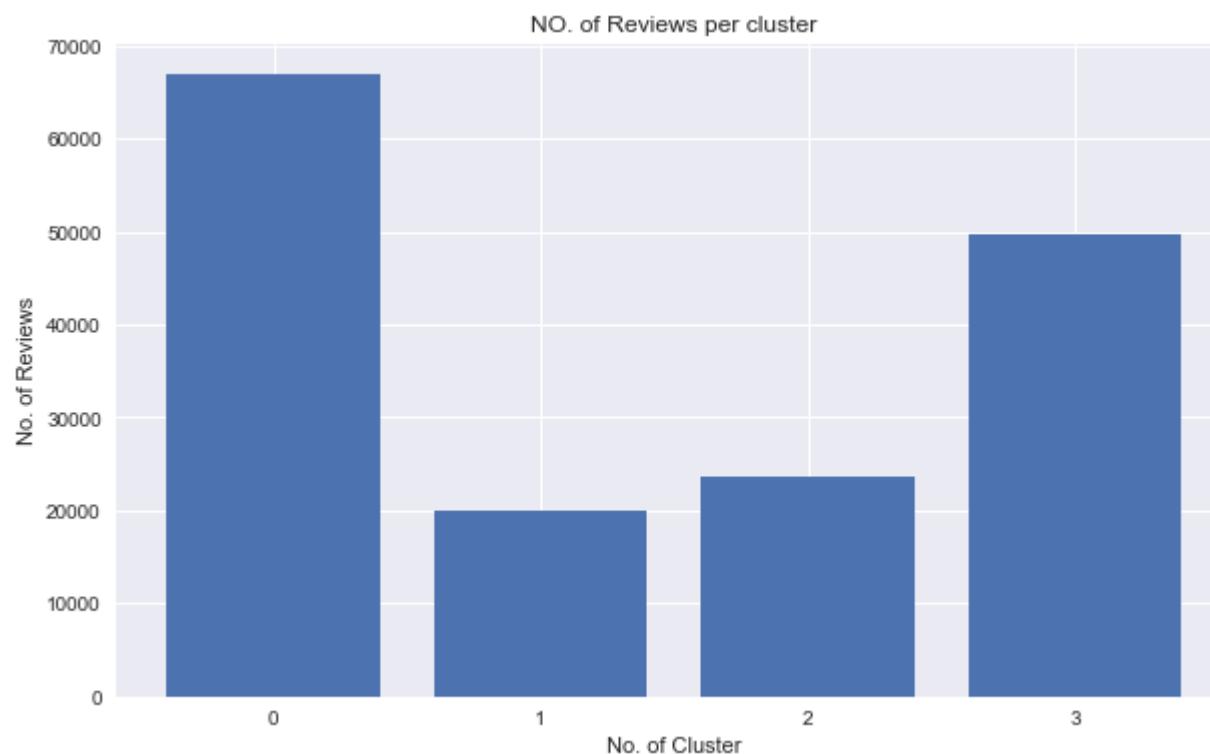
1. from the above plot we observe that after k=4, inertia reduces gradually so we choose k=4 as our knee point

[1.4.2] Review labelling SET 4

In [14]:

```
1 optimal_cluster=4
2 model,cluster_label_df=review_labelling(data,X,optimal_cluster,vect[3],algo[0])
```

```
0    66887
3    49716
2    23635
1    19938
Name: label, dtype: int64
```



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

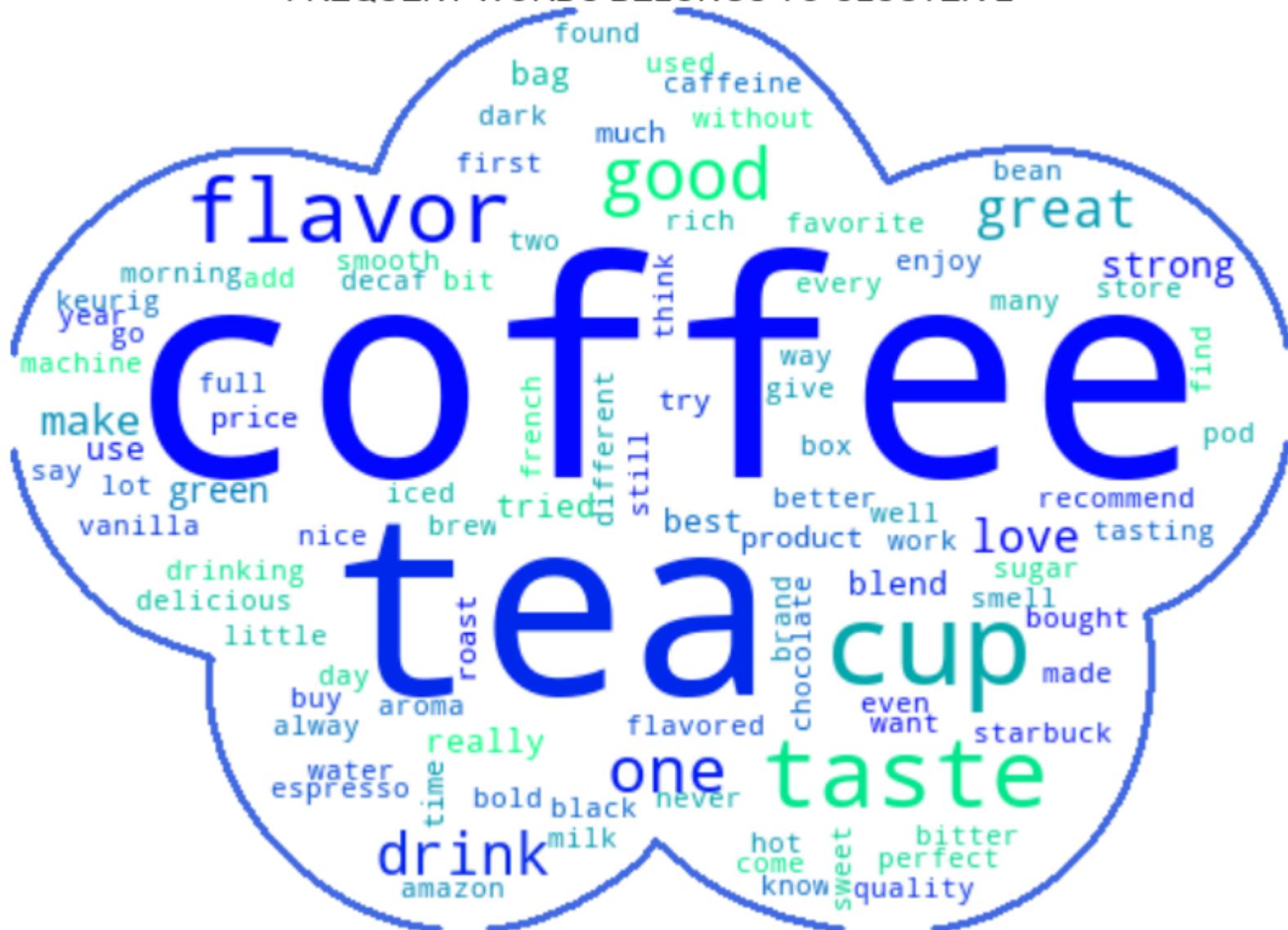
```
In [15]: 1 wordcloud_each_cluster(optimal_cluster,cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1



FREQUENT WORDS BELONGS TO CLUSTER-2



FREQUENT WORDS BELONGS TO CLUSTER-3

**Observation:**

1. From the above wordclouds we cant say anything about cluster because all the clusters are having mixup of words.

[2] Agglomerative Clustering

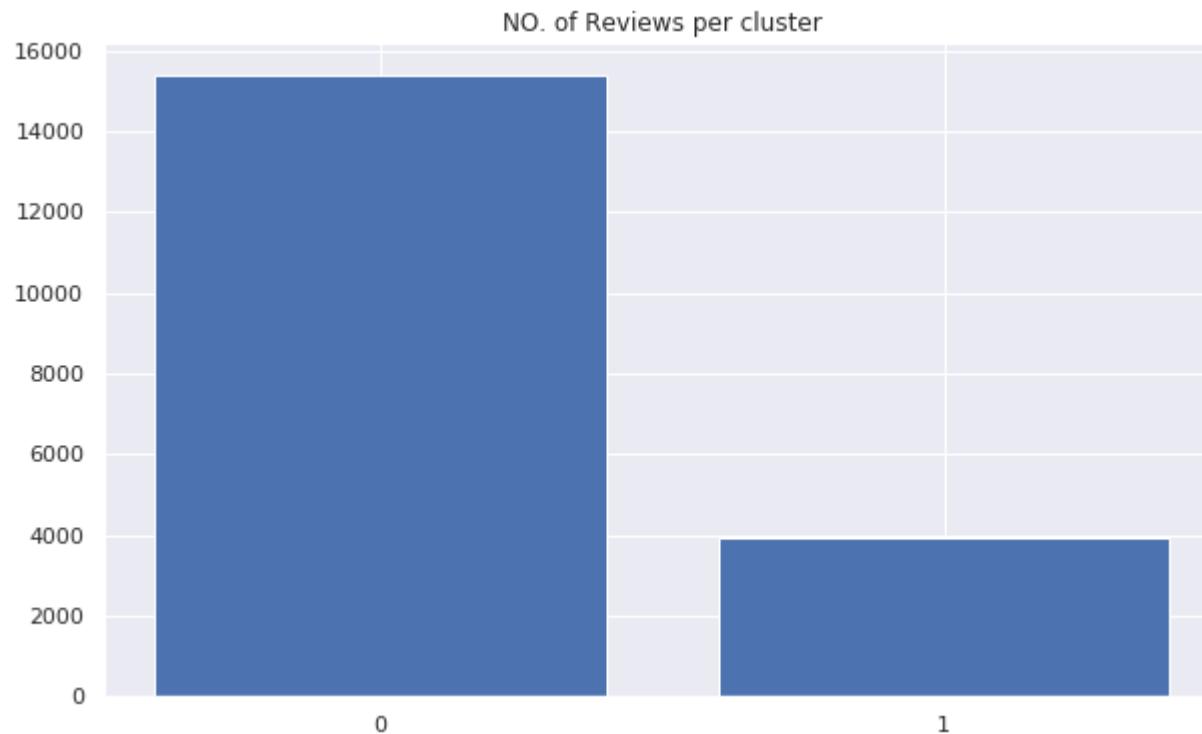
In [14]: 1 no_of_cluster=[2,5]

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

[i] Agglomerative clustering with 2-clusters :

In [11]:

```
1 data=std_data(avg_sent_vectors_agg,mean=True)
2 model,cluster_label_df=review_labelling(data=data,all_Reviews=X_agg,params=no_of_cluster[0],vect=vect[2],alg
0    15406
1    3948
Name: label, dtype: int64
```

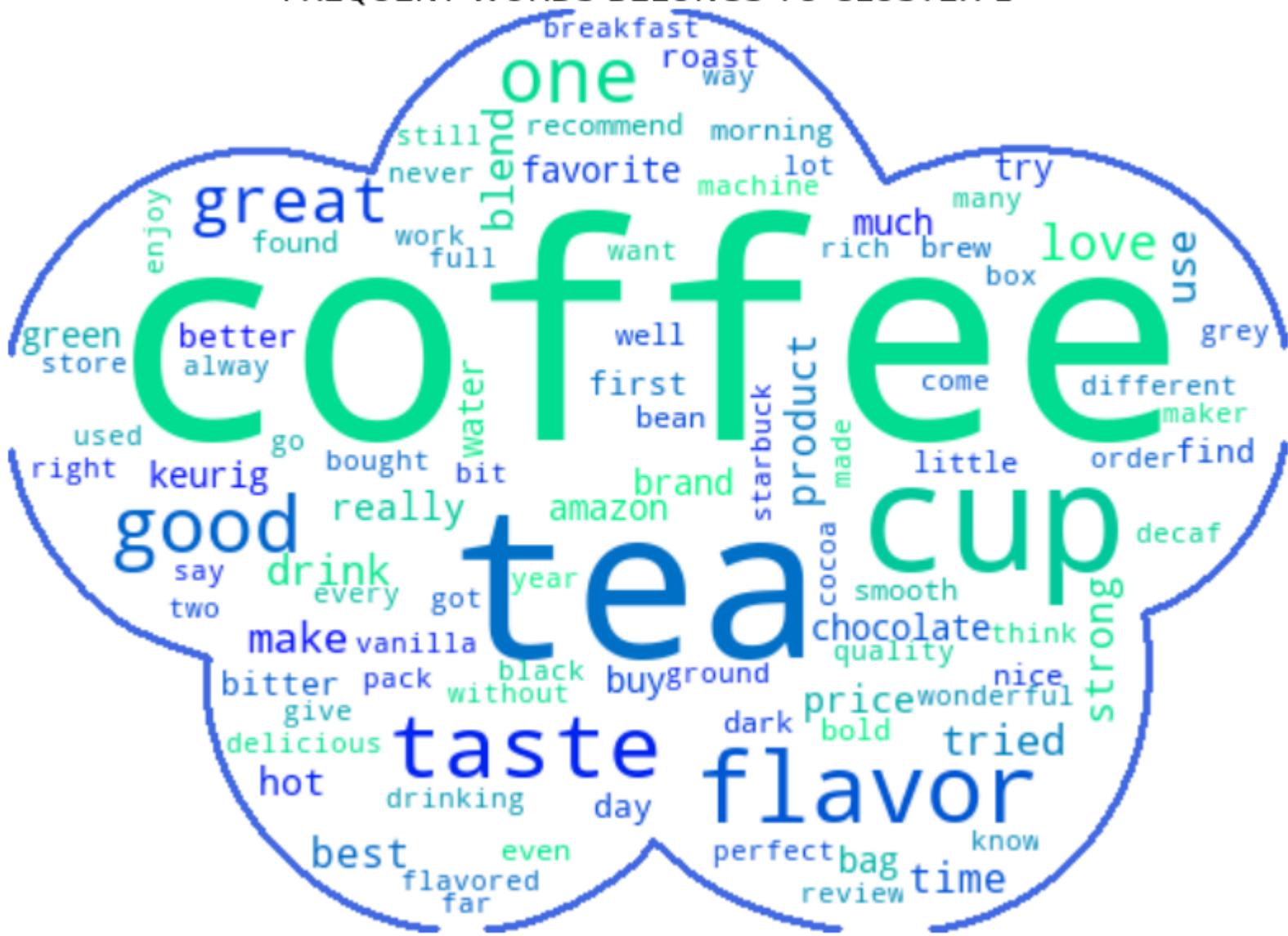


Wordclouds of clusters obtained after applying Agglomerative Clustering on AVG W2V

```
In [12]: 1 wordcloud_each_cluster(no_of_cluster[0],cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1

**Observation:**

- From the above wordclouds we can't say anything about cluster because all the clusters are having mixup of words.

[ii] Agglomerative clustering with 5-clusters :

In [15]:

```
1 data=std_data(avg_sent_vectors_agg,mean=True)
2 model,cluster_label_df=review_labelling(data=data,all_Reviews=X_agg,params=no_of_cluster[1],vect=Vect[2],alg
3 6598
0 4876
1 3948
2 2809
4 1123
Name: label, dtype: int64
```



Wordclouds of clusters obtained after applying Agglomerative Clustering on AVG W2V:

```
In [16]: 1 wordcloud_each_cluster(no_of_cluster[1],cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1



FREQUENT WORDS BELONGS TO CLUSTER-2



FREQUENT WORDS BELONGS TO CLUSTER-3



FREQUENT WORDS BELONGS TO CLUSTER-4

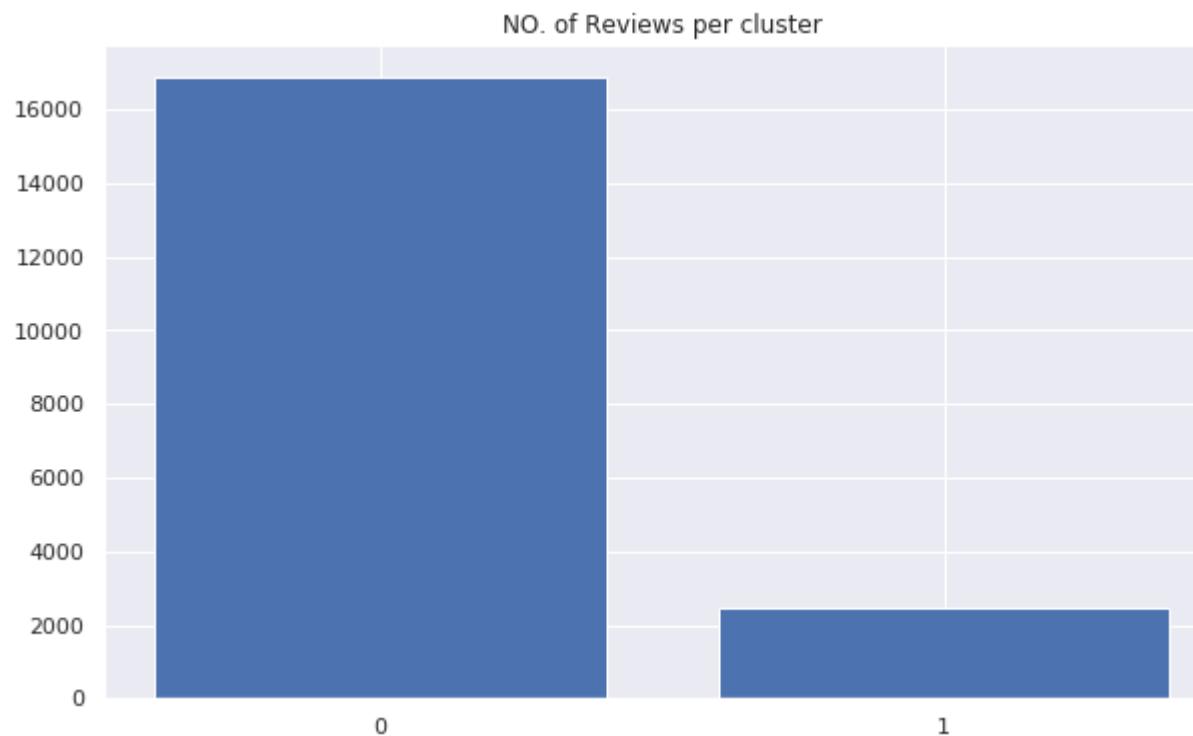
**Observation:**

1. From the above wordclouds we can't say anything about cluster because all the clusters are having mixup of words.

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

[i] Agglomerative clustering with 2-clusters :

```
In [32]: 1 data=std_data(tfidf_sent_vectors_agg,mean=True)
2 model,cluster_label_df=review_labelling(data=data,all_Reviews=X_agg,params=no_of_cluster[0],vect=Vect[3],alg
0    16899
1    2455
Name: label, dtype: int64
```

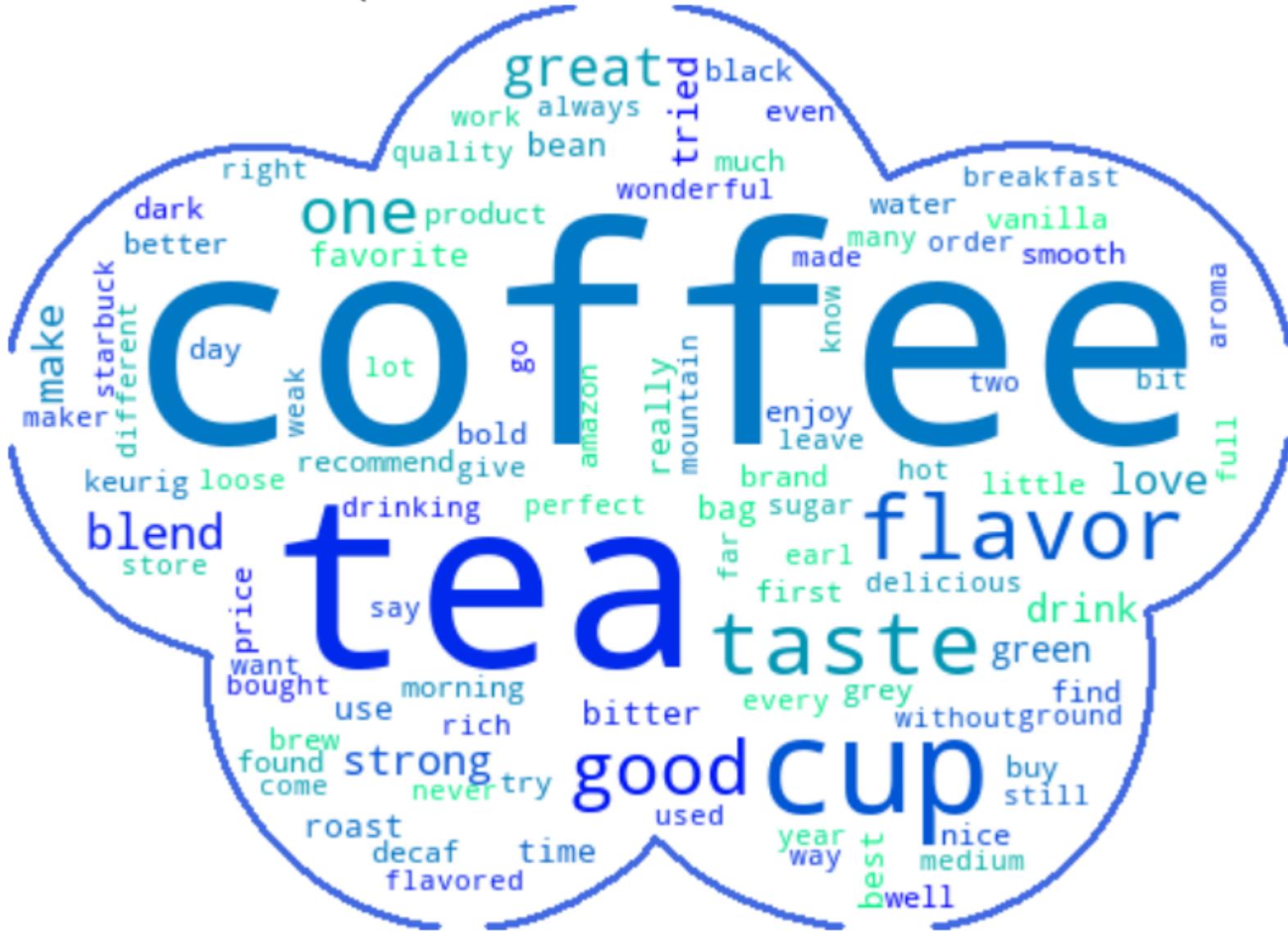


Wordclouds of clusters obtained after applying Agglomerative Clustering on TFIDF W2V:

```
In [33]: 1 wordcloud_each_cluster(no_of_cluster[0],cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1

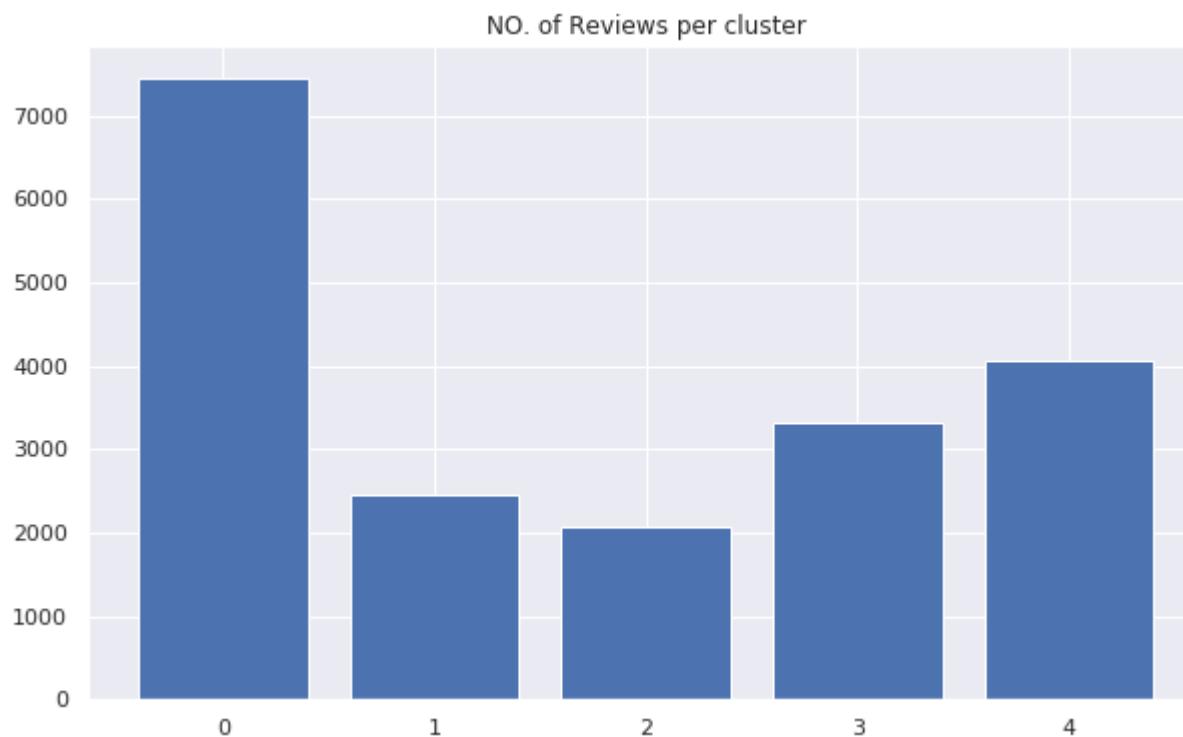
**Observation:**

1. From the above wordclouds we observe that cluster-0 contain mostly positive words(ex great, love, good etc).

[ii] Agglomerative clustering with 5-clusters :

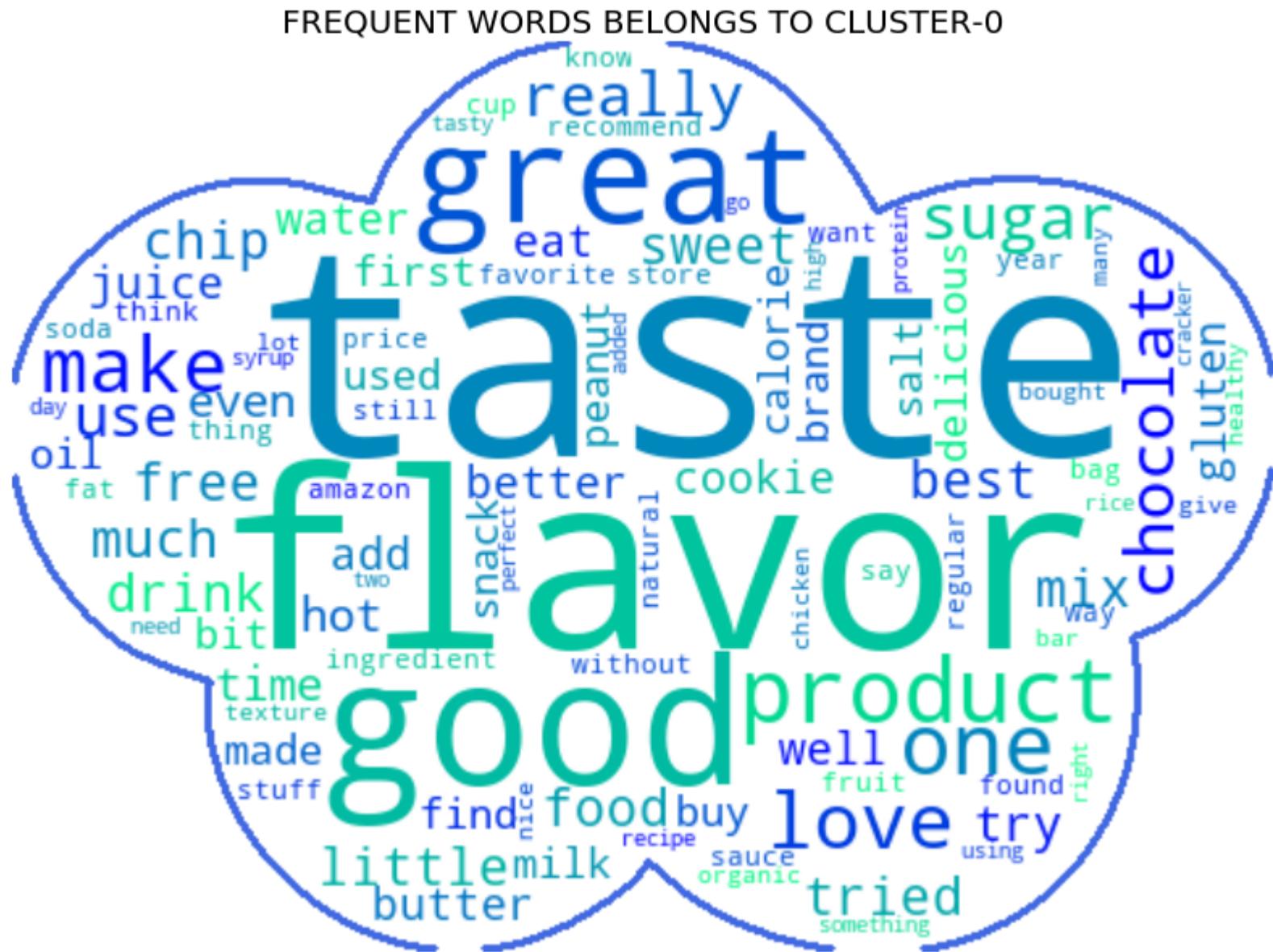
```
In [19]: 1 model,cluster_label_df=review_labelling(data=data,all_Reviews=X_agg,params=no_of_cluster[1],vect=vect[3],alg
```

```
0    7455
4    4067
3    3310
1    2455
2    2067
Name: label, dtype: int64
```

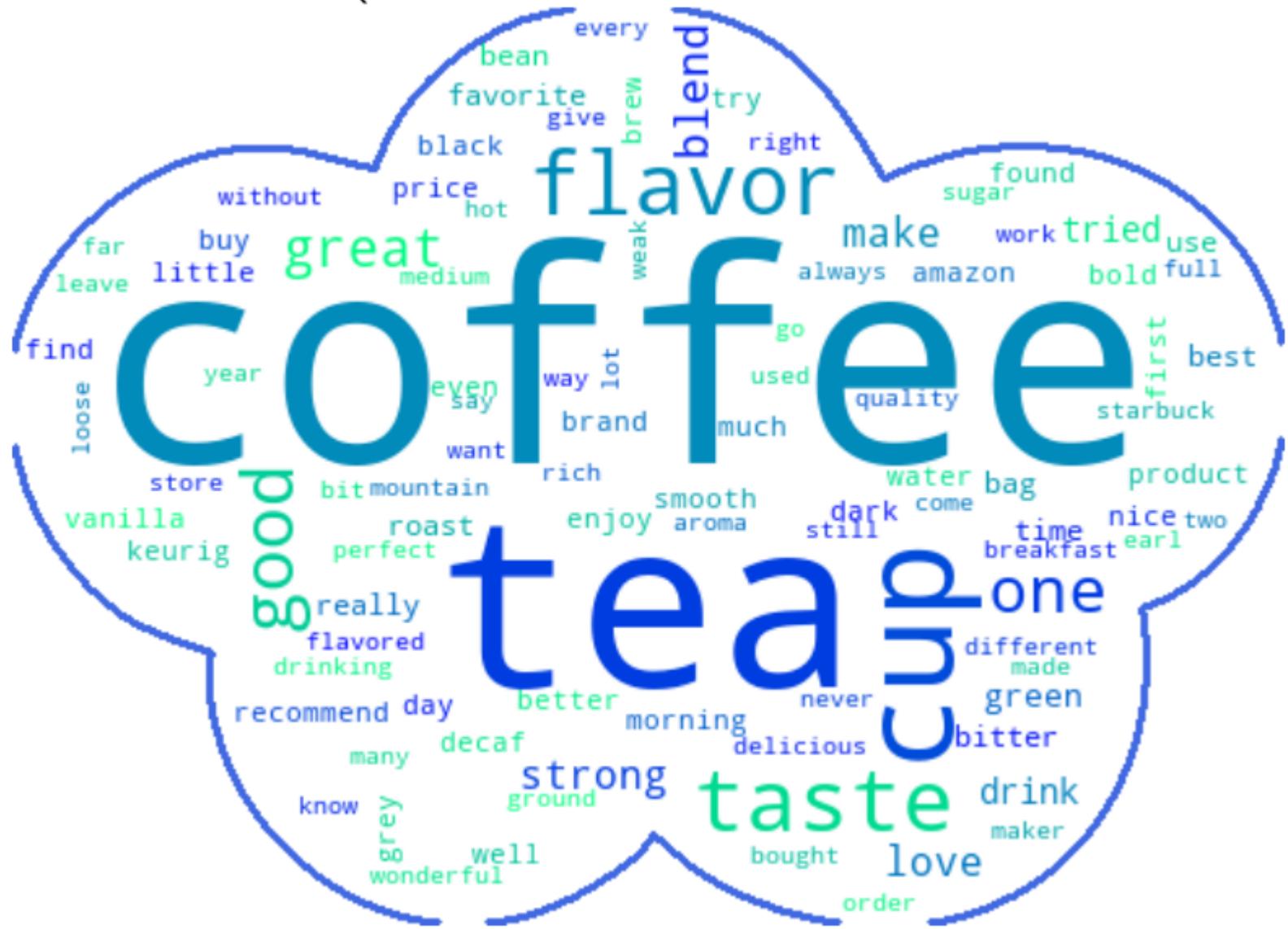


Wordclouds of clusters obtained after applying Agglomerative Clustering on TFIDF W2V:

```
In [20]: 1 wordcloud_each_cluster(no_of_cluster[1],cluster_label_df)
```



FREQUENT WORDS BELONGS TO CLUSTER-1

**Observation:**

- From the above wordclouds we cant say anything about cluster because all the clusters are having mixup of words.

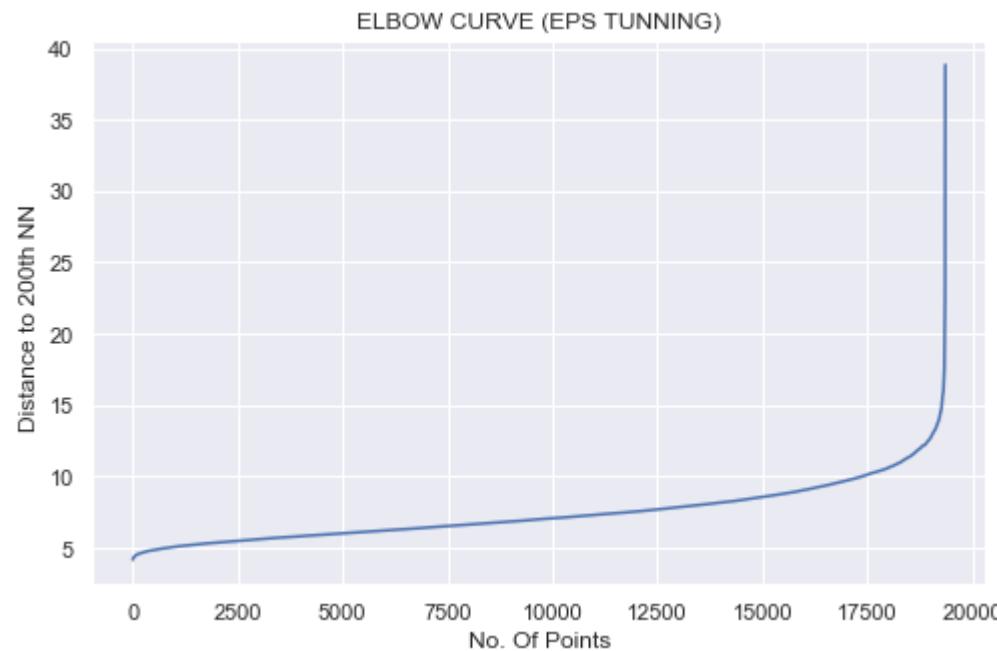
[3] DBSCAN Clustering

[3.1] Applying DBSCAN on AVG W2V, SET 3

[i] Hyperparam Tunning:

In [59]:

```
1 data=std_data(avg_sent_vectors_agg,mean=True)
2 k_dist_plot(data)
```

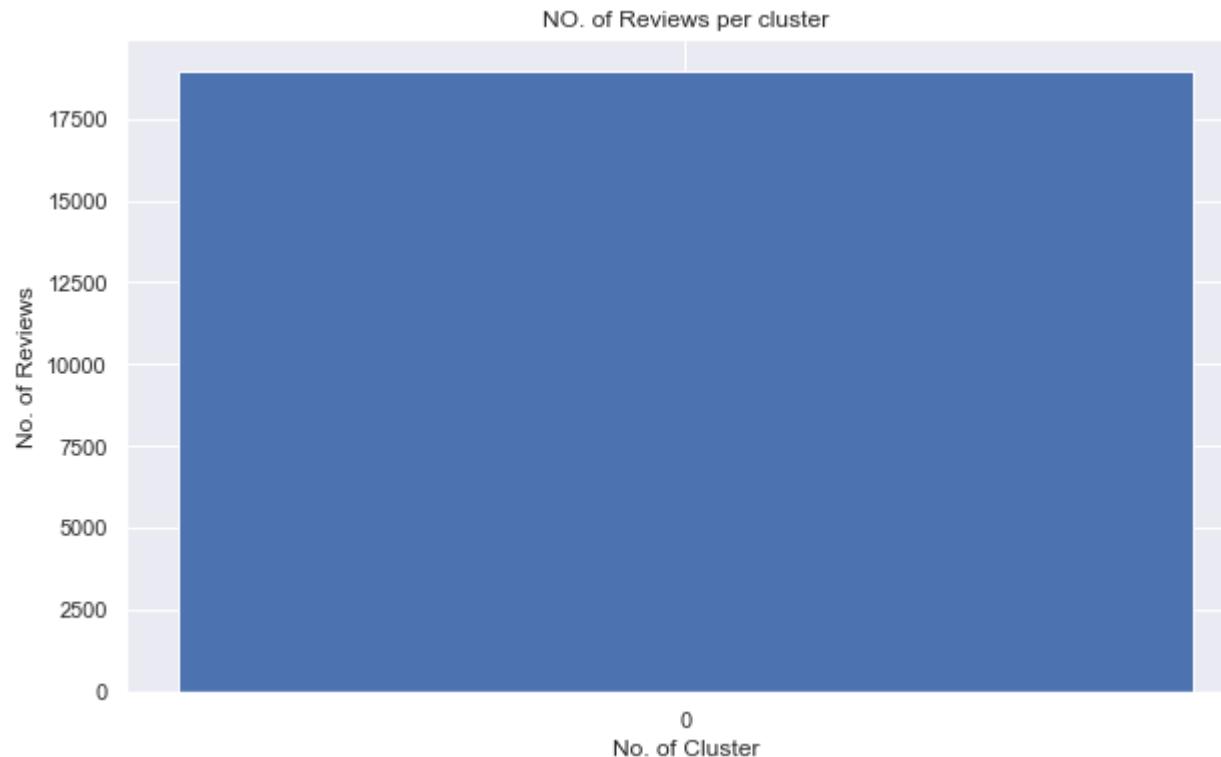


Observation:

1. from the above elbow curve we observe that 15k points have distance to 200th NN are nearly 9. After that distance increases sharply so we select optimal_eps = 9.

In [60]:

```
1 optimal_eps=9
2 model,cluster_label_df=review_labelling(data=data,all_Reviews=X_agg,params=optimal_eps,vect=vect[2],algo=alg)
0    18958
-1      396
Name: label, dtype: int64
```

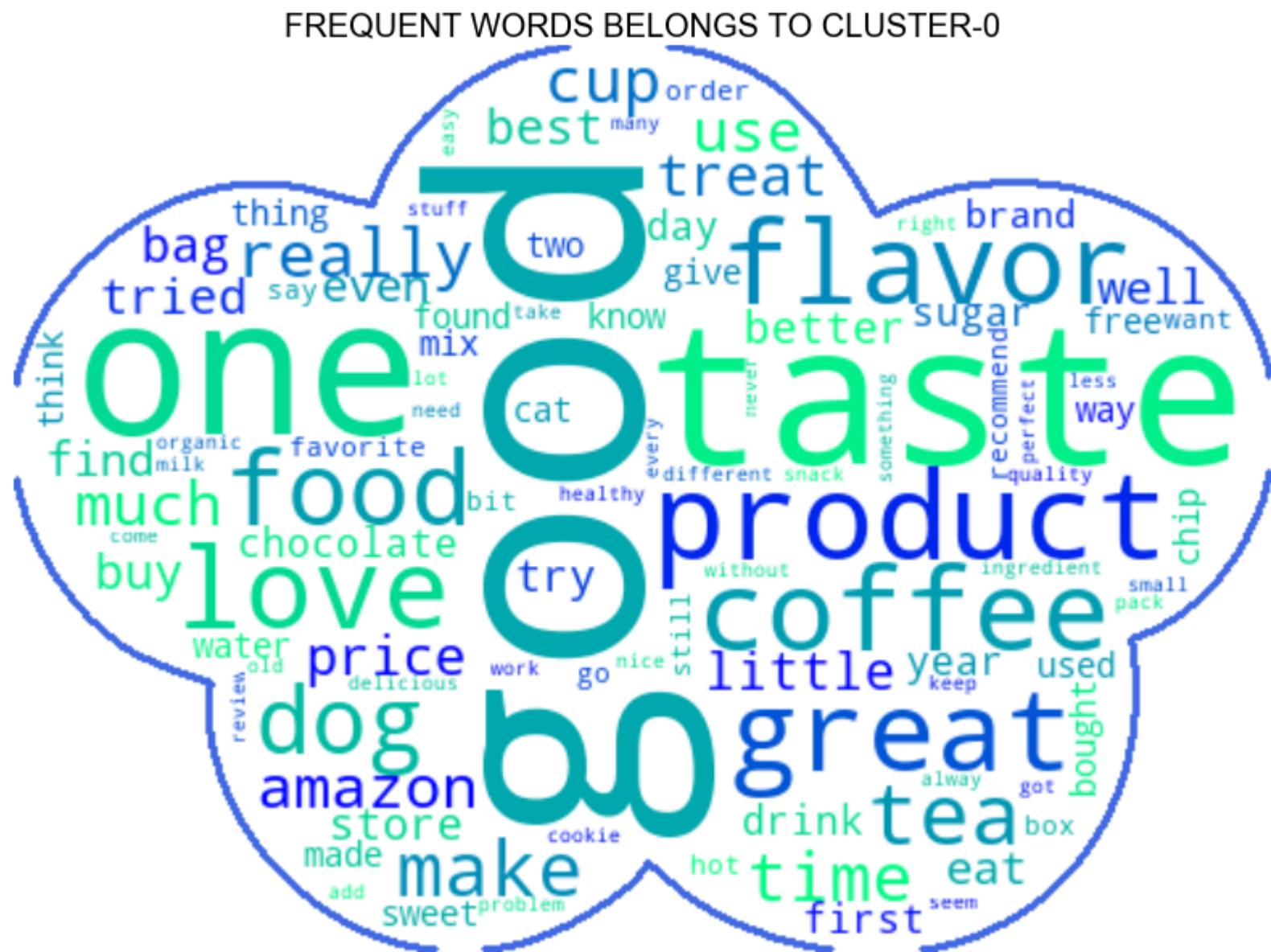


Observation:

1. all the reviews are grouped into one cluster.
2. 396 points are declared as noisy points.

Wordclouds of clusters obtained after applying DBSCAN on AVG W2V

```
In [61]:  
1 n_clusters=1  
2 wordcloud_each_cluster(n_clusters,cluster_label_df)
```



Observation:

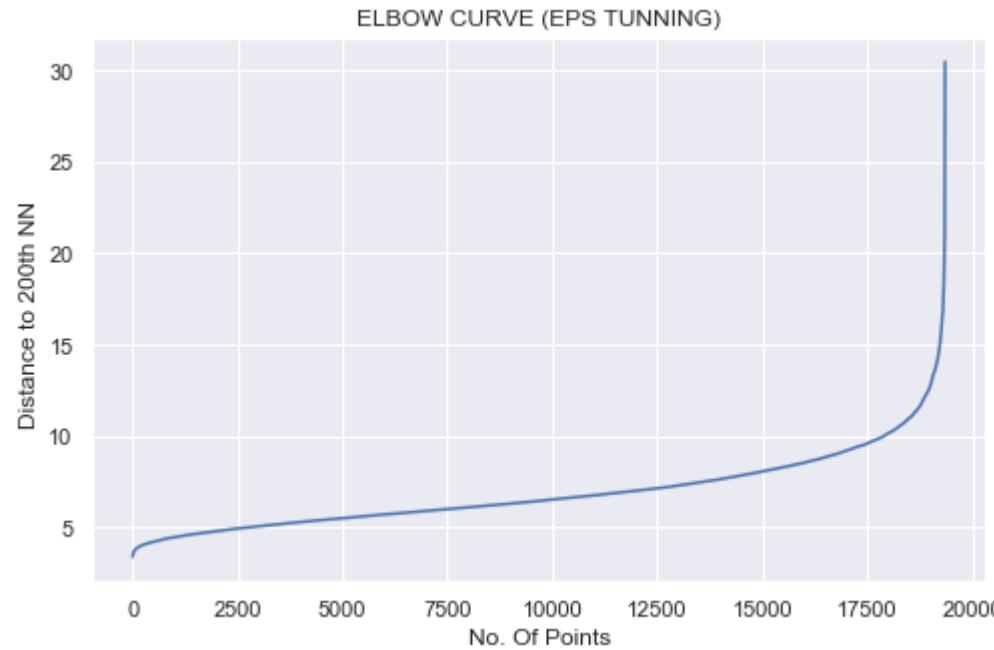
- From the above wordclouds we cant say anything about cluster because all the clusters are having mixup of words.

[3.2] Applying DBSCAN on TFIDF W2V, SET 4

[i] DBSCAN clustering with eps=.5 :

[i] Hyperparam Tuning:

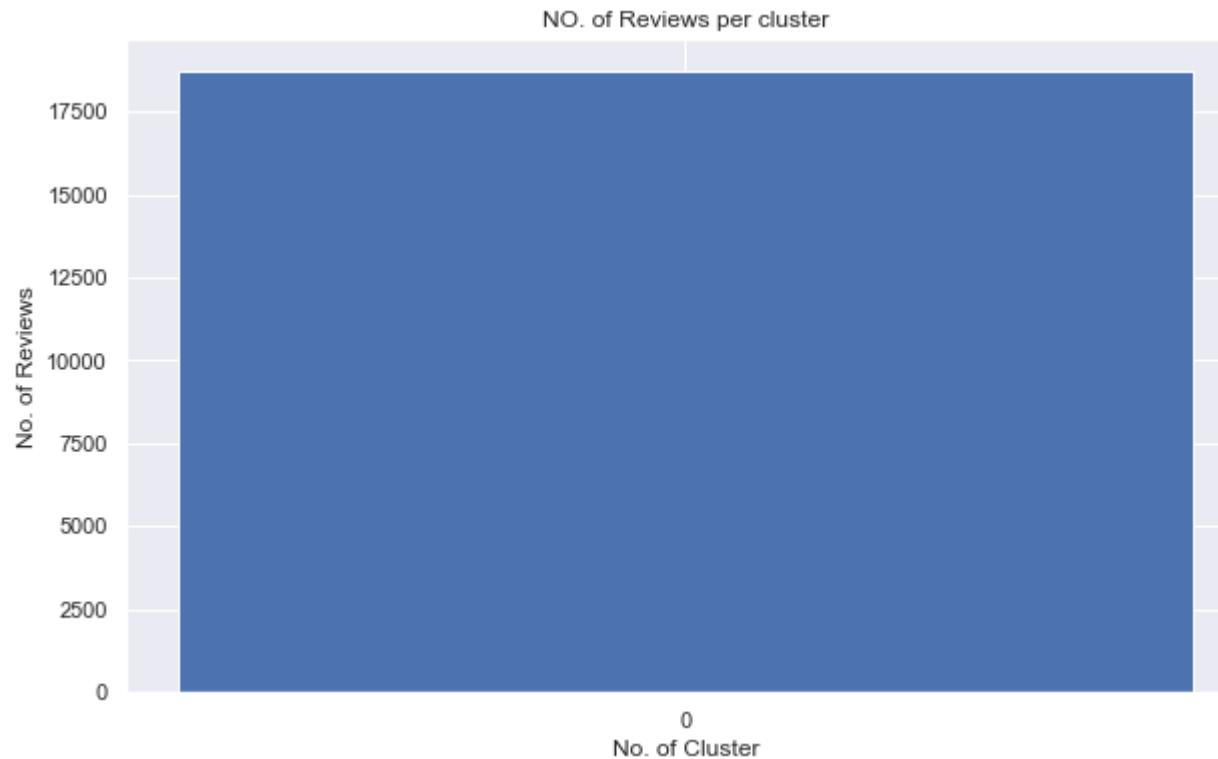
```
In [62]: 1 data=std_data(tfidf_sent_vectors_agg,mean=True)
          2 k_dist_plot(data)
```



Observation:

- from the above elbow curve we observe that 15k points have distance to 200th NN are nearly 8. After that distance increases sharply so we select optimal_eps = 8.

```
In [63]: 1 optimal_eps=8
          2 model,cluster_label_df=review_labelling(data=data,all_Reviews=X_agg,params=optimal_eps,vect=vect[3],algo=alg
0      18732
-1      622
Name: label, dtype: int64
```



Wordclouds of clusters obtained after applying DBSCAN on TFIDF W2V

```
In [43]: 1 n_clusters=1
          2 wordcloud_each_cluster(n_clusters,cluster_label_df)
```

[6] Conclusions

1. K-Means for BoW vectors shows good clustering results because in this technique we observe that our clusters are well separated

one cluster contains positive words and the other one contains supplements and chemical compounds.