# Hierarchical Clustering on Amazon Food Reviews

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews (https://www.kaggle.com/snap/amazon-fine-food-reviews)

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454 Number of users: 256,059 Number of products: 74,258 Timespan: Oct 1999 - Oct 2012 Number of Attributes/Columns in data: 10

Attribute Information:

1. index
2. Id
3. ProductId - unique identifier for the product
4. UserId - unqiue identifier for the user
5. ProfileName
6. HelpfulnessNumerator - number of users who found the review helpful
7. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
8. Score - rating between 1 and 5
9. Time - timestamp for the review
10. Summary - brief summary of the review
11. Text - text of the review
12. ProcessedText - Cleaned & Preprocessed Text of the review

**Objective: Given Amazon Food reviews, convert all the reviews into a vector by taking 5000 data points using three techniques:**
**1. BoW.**
**2. TFIDF.**
**3. Average W2V.**

**Then perform following tasks under each technique:**
**Task 1. Apply Hierarchical Clustering with 5 clusters.**
**Task 2. Apply Hierarchical Clustering with 15 clusters.**
**Task 3. Apply Hierarchical Clustering with 25 clusters.**
**Task 4. Apply Hierarchical Clustering with 35 clusters.**
**Task 5. Apply Hierarchical Clustering with 50 clusters.**

[Q] How to determine if a review is positive or negative?

[Ans] We could use the Score/Rating. A rating of 4 or 5 could be cosnidered a positive review. A review of 1 or 2 could be considered negative. A review of 3 is nuetral and ignored. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

Loading the data

SQLite Database

In order to load the data, We have used the SQLITE dataset as it easier to query the data and visualise the data efficiently. Here as we only want to get the global sentiment of the recommendations (positive or negative), we will purposefully ignore all Scores equal to 3. If the score id above 3, then the recommendation wil be set to "positive". Otherwise, it will be set to "negative".

In [1]:
```
import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
import gensim
from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\gensim\utils.py:862: UserWarning: detected Windows; aliasing chunkize to c
hunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [2]:
```
connection = sqlite3.connect("FinalAmazonFoodReviewsDataset.sqlite")
```

In [3]:
```
data = pd.read_sql_query("SELECT * FROM Reviews", connection)
```

In [4]:
```
data.shape
```

Out[4]:
```
(364171, 12)
```

In [5]:
```
data.head()
```

Out[5]:

| | index | Id | ProductId | UserId | ProfileName | HelpfulnessNumerator | HelpfulnessDenominator | Score | Time | Summary | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | B001E4KFG0 | A3SGXH7AUHU8GW | delmartian | 1 | 1 | Positive | 1303862400 | Good Quality Dog Food | s |
| **1** | 1 | 2 | B00813GRG4 | A1D87F6ZCVE5NK | dll pa | 0 | 0 | Negative | 1346976000 | Not as Advertised | lal |
| **2** | 2 | 3 | B000LQOCH0 | ABXLMWJIXXAIN | Natalia Corres "Natalia Corres" | 1 | 1 | Positive | 1219017600 | "Delight" says it all | cc |
| **3** | 4 | 5 | B006K2ZZ7K | A1UQRSCLF8GW1T | Michael D. Bigham "M. Wassir" | 0 | 0 | Positive | 1350777600 | Great taffy | |
| **4** | 5 | 6 | B006K2ZZ7K | ADT0SRK1MGOEU | Twoapennything | 0 | 0 | Positive | 1342051200 | Nice Taffy | |

In [6]:
```
data["Score"].value_counts()
```

Out[6]:
```
Positive    307061
Negative     57110
Name: Score, dtype: int64
```

In [7]:
```
def changingScores(score):
    if score == "Positive":
        return 1
    else:
        return 0
```

In [8]:
```
# changing score
# Positive = 1
# Negative = 0
actualScore = list(data["Score"])
positiveNegative = list(map(changingScores, actualScore)) #map(function, list of numbers)
data['Score'] = positiveNegative
```

In [9]:
```
data.head()
```

Out[9]:

| | index | Id | ProductId | UserId | ProfileName | HelpfulnessNumerator | HelpfulnessDenominator | Score | Time | Summary | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | B001E4KFG0 | A3SGXH7AUHU8GW | delmartian | 1 | 1 | 1 | 1303862400 | Good Quality Dog Food | I b seve V ca |
| **1** | 1 | 2 | B00813GRG4 | A1D87F6ZCVE5NK | dll pa | 0 | 0 | 0 | 1346976000 | Not as Advertised | Pr a label J S Pea |
| **2** | 2 | 3 | B000LQOCH0 | ABXLMWJIXXAIN | Natalia Corres "Natalia Corres" | 1 | 1 | 1 | 1219017600 | "Delight" says it all | Thi confe tha aro |
| **3** | 4 | 5 | B006K2ZZ7K | A1UQRSCLF8GW1T | Michael D. Bigham "M. Wassir" | 0 | 0 | 1 | 1350777600 | Great taffy | taff |
| **4** | 5 | 6 | B006K2ZZ7K | ADT0SRK1MGOEU | Twoapennything | 0 | 0 | 1 | 1342051200 | Nice Taffy | I wil fo or th |

In [10]:
```
#taking 5000 random samples
data = data.sample(n = 5000)
```

In [11]:
```
data.shape
```

Out[11]:
```
(5000, 12)
```

In [12]:
```
data["Score"].value_counts()
```

Out[12]:
```
1    4211
0     789
Name: Score, dtype: int64
```

In [13]:
```
Data = data
```

In [14]:
```
Data_Labels = data["Score"]
```

In [15]:
```
print(Data.shape)
print(Data_Labels.shape)
```

```
(5000, 12)
(5000,)
```

In [16]:
```
Data.head()
```

Out[16]:

| | index | Id | ProductId | UserId | ProfileName | HelpfulnessNumerator | HelpfulnessDenominator | Score | Time | Summ |
|---|---|---|---|---|---|---|---|---|---|---|
| **280928** | 406309 | 439383 | B00008434F | A3RPL2RYFV2HVZ | J. Kasper "Crazy cat woman" | 0 | 0 | 1 | 1162080000 | My cats re enjoy Au c |
| **347979** | 503037 | 543961 | B006MONQMC | A1V1EP514B5H7Y | asiana | 0 | 0 | 0 | 1339459200 | prefera to p wate think |
| **188179** | 265466 | 287759 | B00032KL1I | A3P86MWNBD0H4K | Ameraida Lomelli | 0 | 0 | 1 | 1326758400 | Great |
| **349521** | 504976 | 546033 | B000EH0RTS | A2OTQ9QOJHXEY9 | Susan Chamberlin "suecalm" | 0 | 0 | 1 | 1233446400 | Excel p |
| **304346** | 438207 | 473879 | B001QXYZ4M | AJLW1DZSHOVGW | Elaine Campbell "Desert Dweller" | 6 | 6 | 1 | 1290902400 | A Mi Sw N Flav Delicic |

# BoW

In [17]:
```
count_vect = CountVectorizer()
Data_BoW = count_vect.fit_transform(Data["ProcessedText"].values)
```

In [18]:
```
print(type(Data_BoW))
print(Data_BoW.shape)
```

```
<class 'scipy.sparse.csr.csr_matrix'>
(5000, 9757)
```

In [19]:
```
#Standardizing our data matrix
Data_BoW_Std = StandardScaler(with_mean = False).fit_transform(Data_BoW)
print(Data_BoW_Std.shape)
print(type(Data_BoW_Std))
```

```
(5000, 9757)
<class 'scipy.sparse.csr.csr_matrix'>

C:\Users\GauravP\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning: Data with input dt
ype int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

# Task 1. Apply Hierarchical Clustering with 5 clusters.

In [20]:
```
clust = AgglomerativeClustering(n_clusters=5, affinity='euclidean')
clust_fit = clust.fit(Data_BoW_Std.toarray())
```

In [21]:
```
Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}

for i in range(5):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))
```

```
Cluster 0 has length 4996
Cluster 1 has length 1
Cluster 2 has length 1
Cluster 3 has length 1
Cluster 4 has length 1
```

In [22]:
```
SilhouetteScore = silhouette_score(Data_BoW_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Silhouette Score = 0.6767017359815775
```

**Silhouette Score of 0.676 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 2. Apply Hierarchical Clustering with 15 clusters.

In [39]:
```
clust = AgglomerativeClustering(n_clusters=15, affinity='euclidean')
clust_fit = clust.fit(Data_BoW_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(15):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_BoW_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 2
Cluster 1 has length 1
Cluster 2 has length 4985
Cluster 3 has length 1
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Silhouette Score = 0.6403668196634671
```

**Silhouette Score of 0.64 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 3. Apply Hierarchical Clustering with 25 clusters.

In [40]:
```
clust = AgglomerativeClustering(n_clusters=25, affinity='euclidean')
clust_fit = clust.fit(Data_BoW_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(25):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_BoW_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 67
Cluster 1 has length 4909
Cluster 2 has length 2
Cluster 3 has length 1
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Cluster 15 has length 1
Cluster 16 has length 1
Cluster 17 has length 1
Cluster 18 has length 1
Cluster 19 has length 1
Cluster 20 has length 1
Cluster 21 has length 1
Cluster 22 has length 1
Cluster 23 has length 1
Cluster 24 has length 1
Silhouette Score = 0.5643914062554746
```

**Silhouette Score of 0.56 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 4. Apply Hierarchical Clustering with 35 clusters.

In [41]:
```
clust = AgglomerativeClustering(n_clusters=35, affinity='euclidean')
clust_fit = clust.fit(Data_BoW_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(35):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_BoW_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 2
Cluster 1 has length 4909
Cluster 2 has length 56
Cluster 3 has length 2
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Cluster 15 has length 1
Cluster 16 has length 1
Cluster 17 has length 1
Cluster 18 has length 1
Cluster 19 has length 1
Cluster 20 has length 1
Cluster 21 has length 1
Cluster 22 has length 1
Cluster 23 has length 1
Cluster 24 has length 1
Cluster 25 has length 1
Cluster 26 has length 1
Cluster 27 has length 1
Cluster 28 has length 1
Cluster 29 has length 1
```

```
Cluster 30 has length 1
Cluster 31 has length 1
Cluster 32 has length 1
Cluster 33 has length 1
Cluster 34 has length 1
Silhouette Score = 0.5578548337794692
```

**Silhouette Score of 0.557 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 5. Apply Hierarchical Clustering with 50 clusters.

In [49]:
```
clust = AgglomerativeClustering(n_clusters=50, affinity='euclidean')
clust_fit = clust.fit(Data_BoW_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(50):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_BoW_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 42
Cluster 1 has length 4909
Cluster 2 has length 1
Cluster 3 has length 2
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Cluster 15 has length 1
Cluster 16 has length 1
Cluster 17 has length 1
Cluster 18 has length 1
Cluster 19 has length 1
Cluster 20 has length 1
Cluster 21 has length 1
Cluster 22 has length 1
Cluster 23 has length 1
Cluster 24 has length 1
Cluster 25 has length 1
Cluster 26 has length 1
Cluster 27 has length 1
Cluster 28 has length 1
Cluster 29 has length 1
```

```
Cluster 30 has length 1
Cluster 31 has length 1
Cluster 32 has length 1
Cluster 33 has length 1
Cluster 34 has length 1
Cluster 35 has length 1
Cluster 36 has length 1
Cluster 37 has length 1
Cluster 38 has length 1
Cluster 39 has length 1
Cluster 40 has length 1
Cluster 41 has length 1
Cluster 42 has length 1
Cluster 43 has length 1
Cluster 44 has length 1
Cluster 45 has length 1
Cluster 46 has length 1
Cluster 47 has length 1
Cluster 48 has length 1
Cluster 49 has length 1
Silhouette Score = 0.5483658373120096
```

**Silhouette Score of 0.548 indicates that clusters are well separated and points are assigned to correct clusters**

# TFIDF

In [42]:
```
tfidf_vect = TfidfVectorizer(ngram_range = (1, 1))
Data_TFIDF = tfidf_vect.fit_transform(Data["ProcessedText"].values)
```

In [43]:
```
print(type(Data_TFIDF))
print(Data_TFIDF.shape)
```

```
<class 'scipy.sparse.csr.csr_matrix'>
(5000, 9576)
```

In [44]:
```
Data_TFIDF_Std = StandardScaler(with_mean = False).fit_transform(Data_TFIDF)
print(Data_TFIDF_Std.shape)
print(type(Data_TFIDF_Std))
```

```
(5000, 9576)
<class 'scipy.sparse.csr.csr_matrix'>
```

# Task 1. Apply Hierarchical Clustering with 5 clusters.

In [56]:
```
clust = AgglomerativeClustering(n_clusters=5, affinity='euclidean')
clust_fit = clust.fit(Data_TFIDF_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(5):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_TFIDF_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 4996
Cluster 1 has length 1
Cluster 2 has length 1
Cluster 3 has length 1
Cluster 4 has length 1
Silhouette Score = 0.6257186943973753
```

**Silhouette Score of 0.625 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 2. Apply Hierarchical Clustering with 15 clusters.

In [57]:
```
clust = AgglomerativeClustering(n_clusters=15, affinity='euclidean')
clust_fit = clust.fit(Data_TFIDF_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(15):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_TFIDF_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 4985
Cluster 1 has length 2
Cluster 2 has length 1
Cluster 3 has length 1
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Silhouette Score = 0.5397361831622534
```

**Silhouette Score of 0.5397 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 3. Apply Hierarchical Clustering with 25 clusters.

In [58]:
```
clust = AgglomerativeClustering(n_clusters=25, affinity='euclidean')
clust_fit = clust.fit(Data_TFIDF_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(25):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_TFIDF_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 3
Cluster 1 has length 2
Cluster 2 has length 4972
Cluster 3 has length 2
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Cluster 15 has length 1
Cluster 16 has length 1
Cluster 17 has length 1
Cluster 18 has length 1
Cluster 19 has length 1
Cluster 20 has length 1
Cluster 21 has length 1
Cluster 22 has length 1
Cluster 23 has length 1
Cluster 24 has length 1
Silhouette Score = 0.506850787992766
```

**Silhouette Score of 0.5068 indicates that clusters are well separated and points are assigned to correct clusters**

# Task 4. Apply Hierarchical Clustering with 35 clusters.

In [59]:
```
clust = AgglomerativeClustering(n_clusters=35, affinity='euclidean')
clust_fit = clust.fit(Data_TFIDF_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(35):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_TFIDF_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 4963
Cluster 1 has length 2
Cluster 2 has length 2
Cluster 3 has length 2
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Cluster 15 has length 1
Cluster 16 has length 1
Cluster 17 has length 1
Cluster 18 has length 1
Cluster 19 has length 1
Cluster 20 has length 1
Cluster 21 has length 1
Cluster 22 has length 1
Cluster 23 has length 1
Cluster 24 has length 1
Cluster 25 has length 1
Cluster 26 has length 1
Cluster 27 has length 1
Cluster 28 has length 1
Cluster 29 has length 1
```

```
Cluster 30 has length 1
Cluster 31 has length 1
Cluster 32 has length 1
Cluster 33 has length 1
Cluster 34 has length 1
Silhouette Score = 0.49217131235867473
```

**Silhouette Score of 0.492 indicates that most of the clusters are well separated most of the and points are assigned to correct clusters**

# Task 5. Apply Hierarchical Clustering with 50 clusters.

In [60]:
```
clust = AgglomerativeClustering(n_clusters=50, affinity='euclidean')
clust_fit = clust.fit(Data_TFIDF_Std.toarray())

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(50):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_TFIDF_Std.toarray(), clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 4951
Cluster 1 has length 1
Cluster 2 has length 1
Cluster 3 has length 1
Cluster 4 has length 1
Cluster 5 has length 1
Cluster 6 has length 1
Cluster 7 has length 1
Cluster 8 has length 1
Cluster 9 has length 1
Cluster 10 has length 1
Cluster 11 has length 1
Cluster 12 has length 1
Cluster 13 has length 1
Cluster 14 has length 1
Cluster 15 has length 1
Cluster 16 has length 1
Cluster 17 has length 1
Cluster 18 has length 1
Cluster 19 has length 1
Cluster 20 has length 1
Cluster 21 has length 1
Cluster 22 has length 1
Cluster 23 has length 1
Cluster 24 has length 1
Cluster 25 has length 1
Cluster 26 has length 1
Cluster 27 has length 1
Cluster 28 has length 1
Cluster 29 has length 1
```

```
Cluster 30 has length 1
Cluster 31 has length 1
Cluster 32 has length 1
Cluster 33 has length 1
Cluster 34 has length 1
Cluster 35 has length 1
Cluster 36 has length 1
Cluster 37 has length 1
Cluster 38 has length 1
Cluster 39 has length 1
Cluster 40 has length 1
Cluster 41 has length 1
Cluster 42 has length 1
Cluster 43 has length 1
Cluster 44 has length 1
Cluster 45 has length 1
Cluster 46 has length 1
Cluster 47 has length 1
Cluster 48 has length 1
Cluster 49 has length 1
Silhouette Score = 0.4726919312584015
```

**Silhouette Score of 0.472 indicates that most of the clusters are well separated most of the and points are assigned to correct clusters**

# AVG W2V

In [51]:
```
i = 0
listOfSentences = []
for sentence in Data["ProcessedText"].values:
    subSentence = []
    for word in sentence.split():
        subSentence.append(word)

    listOfSentences.append(subSentence)
```

In [52]:
```
print(Data['ProcessedText'].values[0])
print("\n")
print(listOfSentences[0:2])
print("\n")
print(type(listOfSentences))
```

```
they are not contain the keurig plastic cup onc you open the packag youll want put them airtight storag but was realli
impress with the smooth flavor and good tast these were littl cheaper than most the keurig kcup ive found was not sure
what expect but these are deff there favorit list


[['they', 'are', 'not', 'contain', 'the', 'keurig', 'plastic', 'cup', 'onc', 'you', 'open', 'the', 'packag', 'youll',
'want', 'put', 'them', 'airtight', 'storag', 'but', 'was', 'realli', 'impress', 'with', 'the', 'smooth', 'flavor', 'an
d', 'good', 'tast', 'these', 'were', 'littl', 'cheaper', 'than', 'most', 'the', 'keurig', 'kcup', 'ive', 'found', 'wa
s', 'not', 'sure', 'what', 'expect', 'but', 'these', 'are', 'deff', 'there', 'favorit', 'list'], ['who', 'has', 'time',
'make', 'your', 'own', 'filter', 'use', 'these', 'for', 'cup', 'the', 'bathroom', 'instead', 'make', 'coffe', 'with',
'them', 'wast', 'money']]


<class 'list'>
```

In [53]:
```
w2vModel = gensim.models.Word2Vec(listOfSentences, size=300, min_count=5, workers=4)
```

In [54]:
```
# compute average word2vec for each review.
sentenceAsW2V = []
for sentence in listOfSentences:
    sentenceVector = np.zeros(300)
    TotalWordsPerSentence = 0
    for word in sentence:
        try:
            vect = w2vModel.wv[word]
            sentenceVector += vect
            TotalWordsPerSentence += 1
        except:
            pass
    if TotalWordsPerSentence!= 0:
        sentenceVector /= TotalWordsPerSentence
        sentenceAsW2V.append(sentenceVector)

print(type(sentenceAsW2V))
print(len(sentenceAsW2V))
print(len(sentenceAsW2V[0]))
```

```
<class 'list'>
5000
300
```

In [55]:
```
Data_W2V_Std = StandardScaler(with_mean = False).fit_transform(sentenceAsW2V)
print(Data_W2V_Std.shape)
print(type(Data_W2V_Std))
```

```
(5000, 300)
<class 'numpy.ndarray'>
```

## Task 1. Apply Hierarchical Clustering with 5 clusters.

In [73]:
```
clust = AgglomerativeClustering(n_clusters=5, affinity='euclidean')
clust_fit = clust.fit(Data_W2V_Std)

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(5):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_W2V_Std, clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 1980
Cluster 1 has length 1266
Cluster 2 has length 465
Cluster 3 has length 529
Cluster 4 has length 760
Silhouette Score = 0.09823567463174446
```

**Silhouette Score of 0.0982 indicates that clusters are overlapped and not well separated**

# Task 2. Apply Hierarchical Clustering with 15 clusters.

In [65]:
```
clust = AgglomerativeClustering(n_clusters=15, affinity='euclidean')
clust_fit = clust.fit(Data_W2V_Std)

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(15):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_W2V_Std, clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 522
Cluster 1 has length 241
Cluster 2 has length 465
Cluster 3 has length 500
Cluster 4 has length 196
Cluster 5 has length 247
Cluster 6 has length 744
Cluster 7 has length 214
Cluster 8 has length 220
Cluster 9 has length 297
Cluster 10 has length 288
Cluster 11 has length 243
Cluster 12 has length 164
Cluster 13 has length 373
Cluster 14 has length 286
Silhouette Score = 0.04030005211897009
```

**Silhouette Score of 0.0403 indicates that clusters are overlapped and not well separated**

# Task 3. Apply Hierarchical Clustering with 25 clusters.

In [66]:
```
clust = AgglomerativeClustering(n_clusters=25, affinity='euclidean')
clust_fit = clust.fit(Data_W2V_Std)

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(25):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_W2V_Std, clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 744
Cluster 1 has length 220
Cluster 2 has length 295
Cluster 3 has length 122
Cluster 4 has length 172
Cluster 5 has length 243
Cluster 6 has length 147
Cluster 7 has length 170
Cluster 8 has length 187
Cluster 9 has length 297
Cluster 10 has length 60
Cluster 11 has length 125
Cluster 12 has length 164
Cluster 13 has length 373
Cluster 14 has length 286
Cluster 15 has length 169
Cluster 16 has length 304
Cluster 17 has length 158
Cluster 18 has length 69
Cluster 19 has length 196
Cluster 20 has length 49
Cluster 21 has length 26
Cluster 22 has length 169
Cluster 23 has length 92
Cluster 24 has length 163
Silhouette Score = 0.038586709442794166
```

**Silhouette Score of 0.0385 indicates that clusters are overlapped and not well separated**

# Task 4. Apply Hierarchical Clustering with 35 clusters.

In [67]:
```
clust = AgglomerativeClustering(n_clusters=35, affinity='euclidean')
clust_fit = clust.fit(Data_W2V_Std)

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(35):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_W2V_Std, clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 147
Cluster 1 has length 174
Cluster 2 has length 286
Cluster 3 has length 187
Cluster 4 has length 139
Cluster 5 has length 125
Cluster 6 has length 373
Cluster 7 has length 169
Cluster 8 has length 158
Cluster 9 has length 196
Cluster 10 has length 60
Cluster 11 has length 163
Cluster 12 has length 182
Cluster 13 has length 89
Cluster 14 has length 329
Cluster 15 has length 72
Cluster 16 has length 304
Cluster 17 has length 62
Cluster 18 has length 69
Cluster 19 has length 253
Cluster 20 has length 49
Cluster 21 has length 26
Cluster 22 has length 169
Cluster 23 has length 92
Cluster 24 has length 145
Cluster 25 has length 164
Cluster 26 has length 76
Cluster 27 has length 121
Cluster 28 has length 50
Cluster 29 has length 33
```

```
Cluster 30 has length 25
Cluster 31 has length 82
Cluster 32 has length 44
Cluster 33 has length 233
Cluster 34 has length 154
Silhouette Score = 0.04050277449371593
```

**Silhouette Score of 0.0405 indicates that clusters are overlapped and not well separated**

# Task 5. Apply Hierarchical Clustering with 50 clusters.

In [68]:
```
clust = AgglomerativeClustering(n_clusters=50, affinity='euclidean')
clust_fit = clust.fit(Data_W2V_Std)

Cluster_indices = {i: np.where(clust.labels_ == i) for i in range(clust.n_clusters)}
for i in range(50):
    length = len(Cluster_indices[i][0])
    print("Cluster "+str(i)+" has length "+str(length))

SilhouetteScore = silhouette_score(Data_W2V_Std, clust.labels_)
print("Silhouette Score = "+str(SilhouetteScore))
```

```
Cluster 0 has length 162
Cluster 1 has length 60
Cluster 2 has length 182
Cluster 3 has length 69
Cluster 4 has length 196
Cluster 5 has length 164
Cluster 6 has length 84
Cluster 7 has length 108
Cluster 8 has length 139
Cluster 9 has length 253
Cluster 10 has length 26
Cluster 11 has length 146
Cluster 12 has length 76
Cluster 13 has length 89
Cluster 14 has length 33
Cluster 15 has length 72
Cluster 16 has length 233
Cluster 17 has length 143
Cluster 18 has length 52
Cluster 19 has length 77
Cluster 20 has length 49
Cluster 21 has length 197
Cluster 22 has length 169
Cluster 23 has length 81
Cluster 24 has length 145
Cluster 25 has length 107
Cluster 26 has length 93
Cluster 27 has length 121
Cluster 28 has length 50
Cluster 29 has length 71
```

```
Cluster 30 has length 25
Cluster 31 has length 82
Cluster 32 has length 44
Cluster 33 has length 59
Cluster 34 has length 154
Cluster 35 has length 62
Cluster 36 has length 63
Cluster 37 has length 133
Cluster 38 has length 66
Cluster 39 has length 41
Cluster 40 has length 132
Cluster 41 has length 98
Cluster 42 has length 44
Cluster 43 has length 84
Cluster 44 has length 87
Cluster 45 has length 72
Cluster 46 has length 171
Cluster 47 has length 51
Cluster 48 has length 70
Cluster 49 has length 15
Silhouette Score = 0.04023371899154547
```

**Silhouette Score of 0.0402 indicates that clusters are overlapped and not well separated**