Program 3 – Text Clustering

Poojitha Amin
SJSU ID: 011811306

Rank – 9
NMI-Score - 0.4704

1. Approach

DBSCAN pseudocode

```
/*DBSCAN pseudocode*/
/* Input:
Data – Input array
Epsilon – Distance
MinSamples – Minimum number of connected points
*/
DBSCAN_Implementation(Data, epsilon, MinSamples){
    output = Output List of length-Data, with initial values 0
    clusterNo = 0
    for each point in the range of 0 and Length of Data{
            if output(point)=undefined {
                    Get ConnectedComponents = getConnectedComponents(X, Point,
epsilon, output)
            if length(ConnectedComponents) < MinSamples{
                    output(point) = noise
            else
                    Increment clusterNo
                    connectClusters(Data, epsilon, MinSamples, point, output, clusterNo,
ConnectedComponents)
    }
    }
    return output
    }

/* Calculate distance between a given point and other points*/
/* Input:
Data – Input array
Output – output cluster labels
Point – Core point index
Epsilon – Distance
*/
```

```
    getConnectedComponents(Data, epsilon, Point, output){
        ConnectedComponents = []
        for each newPoint in the range of 0 and Length of Data{
                if output(newPoint) = undefined or output(newPoint) = noise {
                        Calculate distance between newPoint and Point
                        If distance < epsilon:
                                Add newPoint to ConnectedComponents
                    }
        Return ConnectedComponents
}
```

/*Connect neighboring clusters*/

/ Input:*
Data – Input array
Output – output cluster labels
Point – Core point index
ConnectedComponents – Core point neighbours
Epsilon – Distance
MinSamples – Minimum number of connected points
**/*

```
    connectClusters(Data, output, Point, ConnectedComponents, clusterNo, epsilon,
MinSamples) {
        index = 0
        output(point) = clusterNo
        for index in the range of (0 and len(ConnectedComponents)){
                    val = ConnectedComponents(idx)
                    if val is set as noise point {
                        Assign clusterNo to it
                    }
                    Else if output(val) is undefined{
                        Assign clusterNo to it
                        Get NewConnectedComponents = getConnectedComponents(Date, Val,
    epsilon, output)
                        if length(ConnectedComponents) < MinSamples{
                                Add NewConnectedComponents to ConnectedComponents
                        }
                   }
        Increment index

    }
```

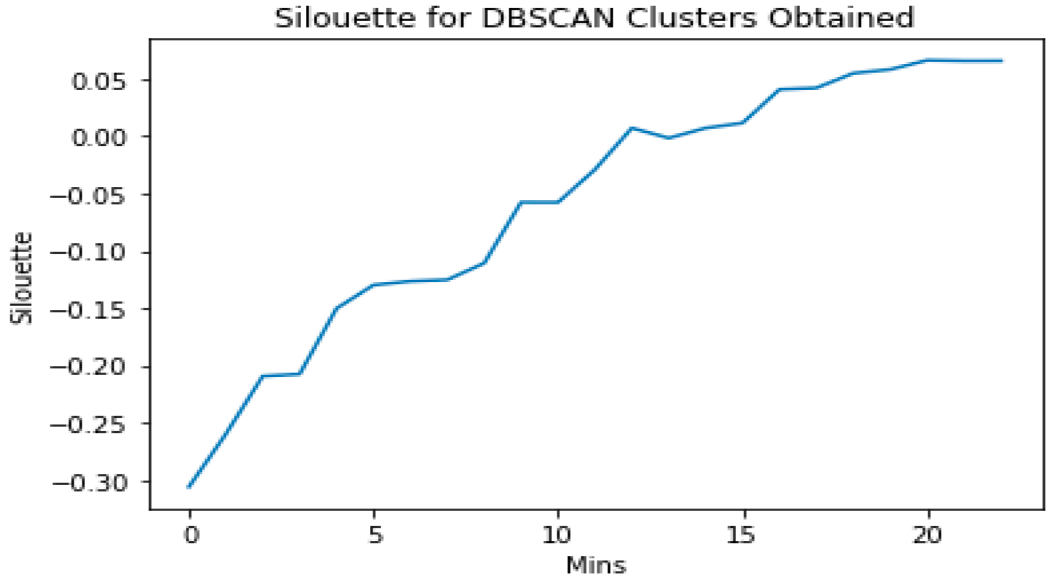
2. Eps values for the Minpoints range 3 to 21.

Made use of the R function – kNNdistplot, to plot the k-distance graph for the k-nearest neighbor distances for the given input in the ascending order. This calculates the average of the distances of every point to its nearest neighbors. The value of k corresponds to the MinPoints. From the knee point in the plot, the approximate eps value was identified for each value of Minpoint.

MinPoints	Epsilon
3	0.28
5	0.28
7	0.3
9	0.3
11	0.3
13	0.3
15	0.3
17	0.3
19	0.3
21	0.4
23	0.4

3. Internal Evaluation metric for the clusters

Made use of the Silhouette Coefficient as the internal metric to evaluate the clusters created. It is calculated using the mean intra cluster distance and the mean nearest cluster distance for each sample. The Silhouette Coefficient was plotted for all the clusters derived by using the epsilon and MinPoints values listed above.



We can see that the number of MinPoints=19 has the near highest Silhouette Coefficient. Hence, epsilon 0.3 and MinPoints=19 should be giving us the optimal number of clusters.

4. Other Pre-Processing Steps Used

TF-IDF

Term Frequency, Inverse Document Frequency, is a way to score the importance of the words based on its frequency across all the documents. If a word appears in many documents TF-IDF transformer gives it a lower score and vice-versa. Applied this technique assuming this would help in better clustering of the text entries.

```
from sklearn.feature_extraction.text import TfidfTransformer
transformer = TfidfTransformer()
```

Dimensionality Reduction – PCA

In order to reduce the number of dimensions used in the model development, PCA was applied on the dataset. It finds a lower dimensional linear subspace that the data is confined to.

```
from sklearn.decomposition import KernelPCA
pca = KernelPCA(n_components=150, kernel='linear')
```

Also tried Random Projection dimensionality reduction method, but it did not help in optimizing the solution.

Normalization

Normalized the dataset in order to deal with the data of varying scales and to have a fair comparison among points in the cluster. Normalization scales the data in the range of [0,1].

```
X_normalized = preprocessing.normalize(X_New, norm='l2')
```

Distance Measures

Tried multiple distance measures to determine the clusters in the dataset such as Euclidean distance and Cosine distance.