

EECS 435 Data Mining Project 2 Report

Sinan He
Computer Science
Case Western Reserve
University
Cleveland, Ohio
sxh827@case.edu

Yanbo Huang
Computer Science
Case Western Reserve
University
Cleveland, Ohio
yxh1050@case.edu

Yunzhou Cao
Computer Science
Case Western Reserve
University
Cleveland, Ohio
yxc1426@case.edu

ABSTRACT

In this project, we use 3 different clustering methods to perform experiments. Each method is analyzed by two different distance measures.

1.1 Methods

1.1.1 Feature vectors

We use the Bag-Of-Words approach to convert the raw data into feature vectors. To be specific, function CountVectorizer from scikit-learn. What's more, we only the third column separated by “|” from the given data.

1.1.2 Distance Measures

We use two different distance measures: Euclidean and Cosine.

(i) Euclidean Distance

The reason why we choose Euclidean is that it is the most commonly used and straightforward measure. We want to use it to compare it with other measures.

Figure 1 shows the Euclidean Distance Histogram.

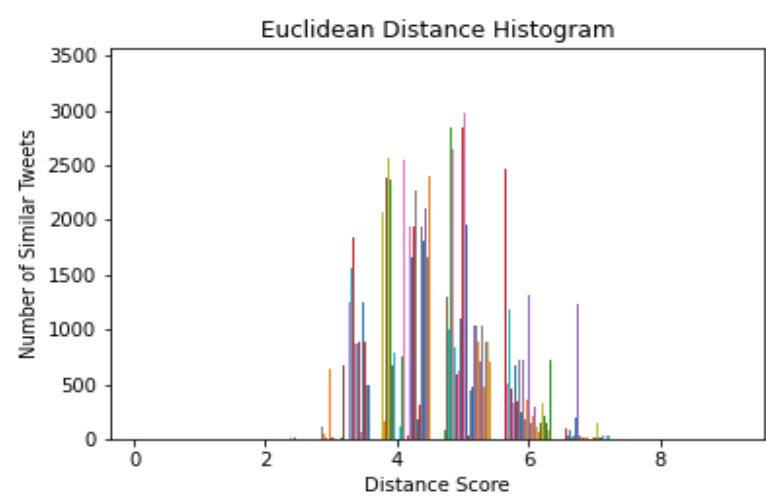


Figure 1. Euclidean Distance Histogram

(ii) Cosine distance

We choose Cosine distance because it works better when there are plenty of feature vectors. Also, this measure focuses more on the difference of directions

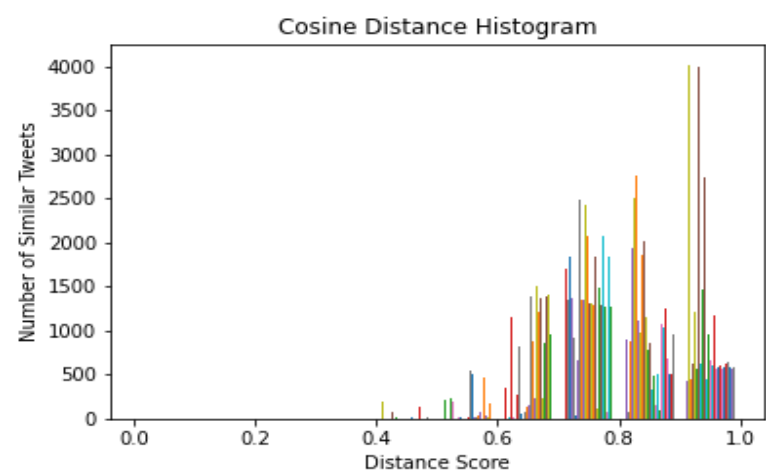


Figure 2. Cosine Distance Histogram

1.1.3 Clustering Algorithm

(i) K-means

We used scikit-learn and nltk to implement K-means clustering. To optimize the number of clustering, we implemented an elbow method using seaborn library to plot and determine parameter n_clusters for each distance. Given that the function KMeans in sklearn only can use Euclidean distance, we used `nltk.cluster.kmeans.KMeansClusterer` for Cosine distance. The optimal number of clusters is 4 for Euclidean and 5 for Cosine. We set `init='k-means++'` so that we don't need to initialize the mean by ourselves.

(ii) DBSCAN

We used scikit-learn and nltk to implement dbscan clustering. We use the two main parameters for DBSCAN are the minimum number of points that constitute a cluster (minPts) and the size of the neighborhood (eps). As a rule of thumb, it is best to set minPts to at least the number of features in your data. Also, the optimized value for eps can be determined by a k-distance graph, but a small value is always preferred. The optimal number of clusters for Euclidean is 17, for Cosine is 21.

(iii) Agglomerative Clustering

This method is implemented with scikit-learn. In the function `AgglomerativeClustering`, we set the parameters affinity to be "precomputed" and linkage to be "complete" since we have computed the distance matrix to draw the histogram in the previous section. Also, the optimal number of clusters is 4 for Euclidean and 3 for Cosine since we tried the number of clusters from 2 to 6.

1.2 Data

Based on the feature vectors, we are able to summary some basic information of the data in the table below.

#Documents	#Term Tokens	#Unique Terms	Avg Term/ Document
4061	49531	1546	12.2

Table 1. Basic Statistics for Datasets

1.3 Results

1.3.1. Performance

(i) K-means

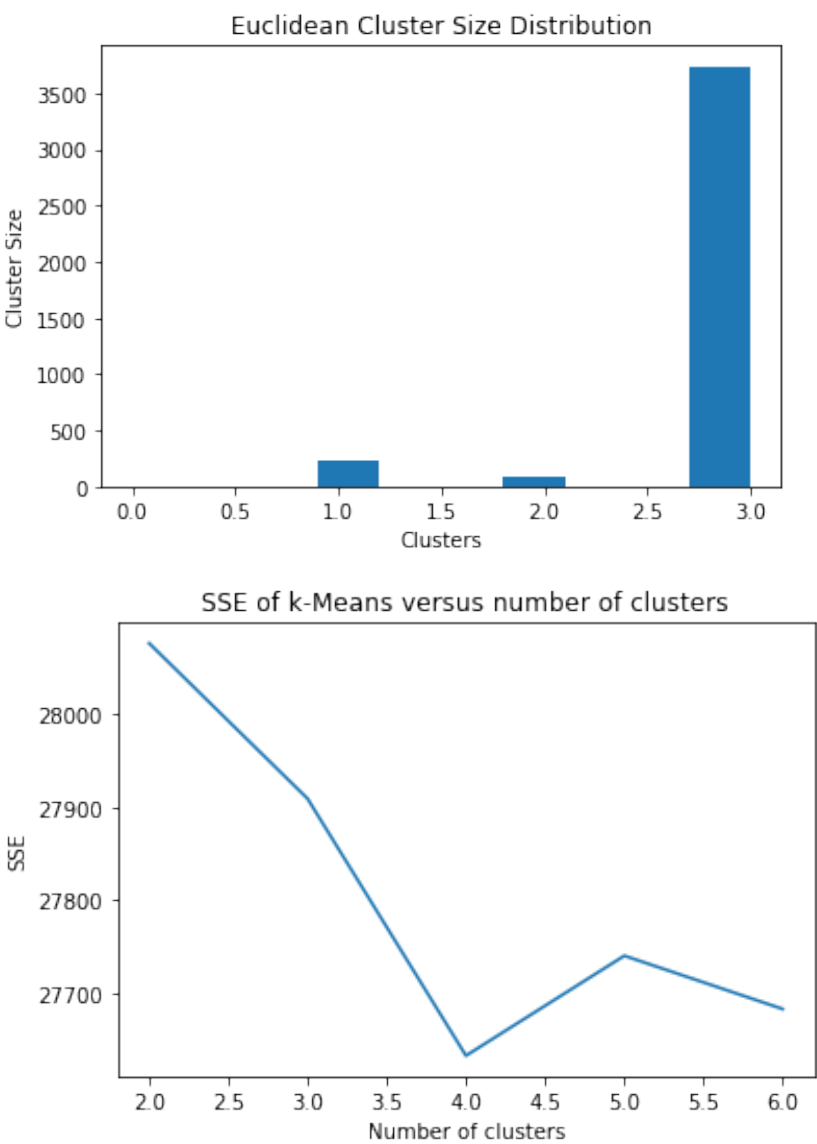


Figure 1. Size distribution and elbow method of K-means(Euclidean)

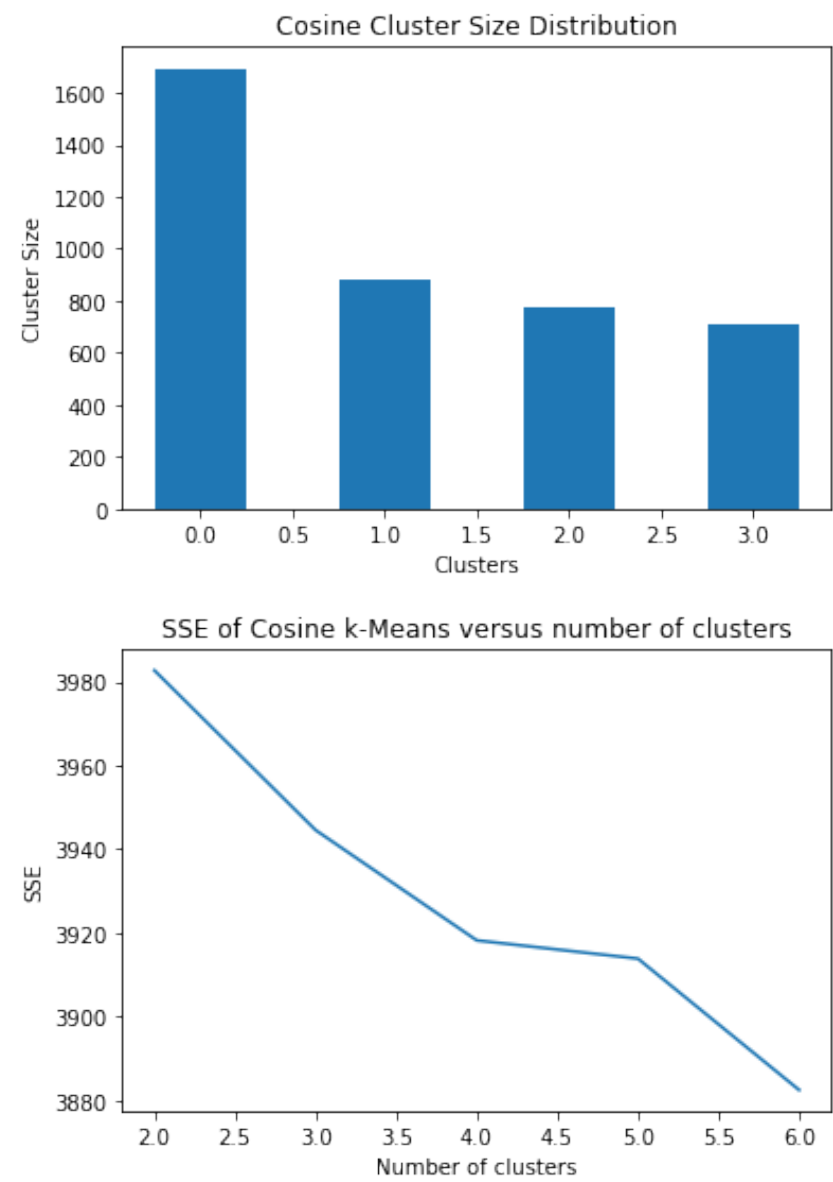


Figure 2. Size distribution and elbow method of K-means(Cosine)

(ii) DBSCAN

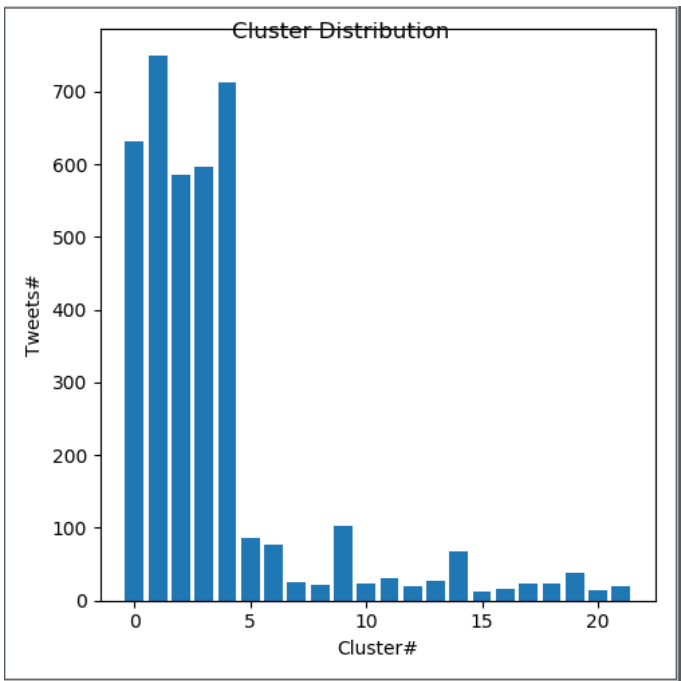


Figure 3. Size distribution of dbscan(Cosine)

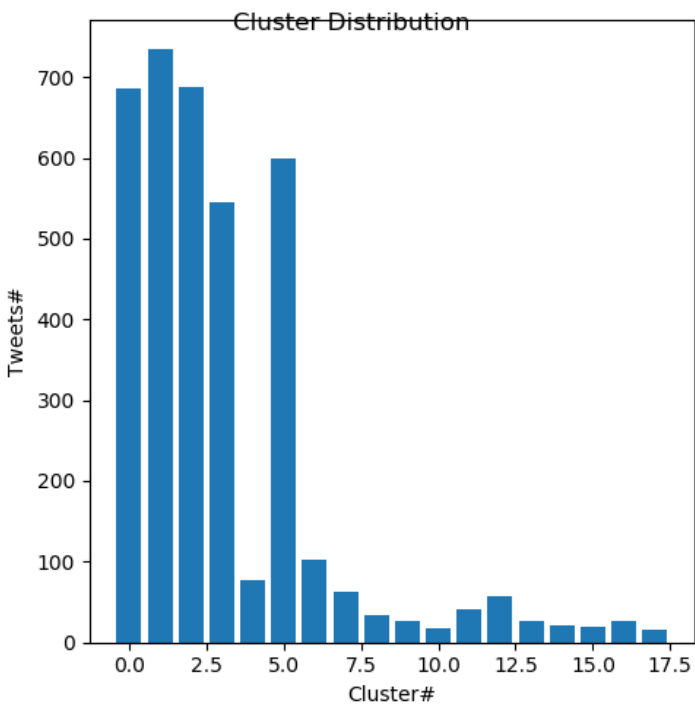


Figure 4. Size distribution of dbscan(Euclidean)

(iii) Agglomerative Clustering

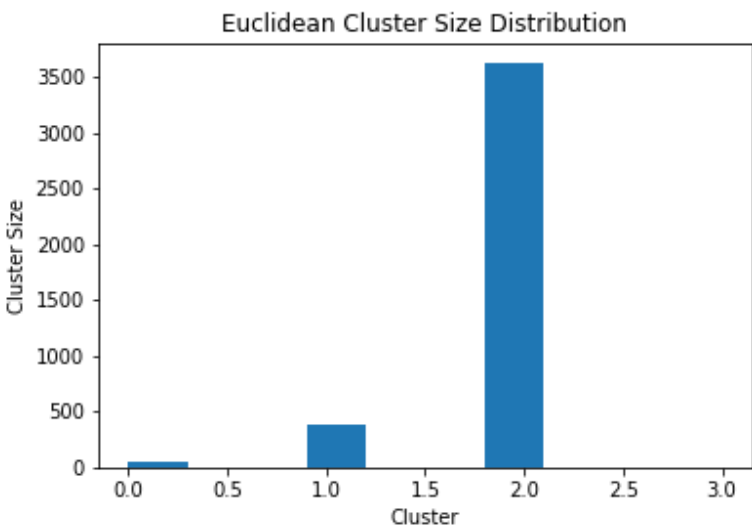


Figure 5. Size distribution of Agglomerative(Euclidean)

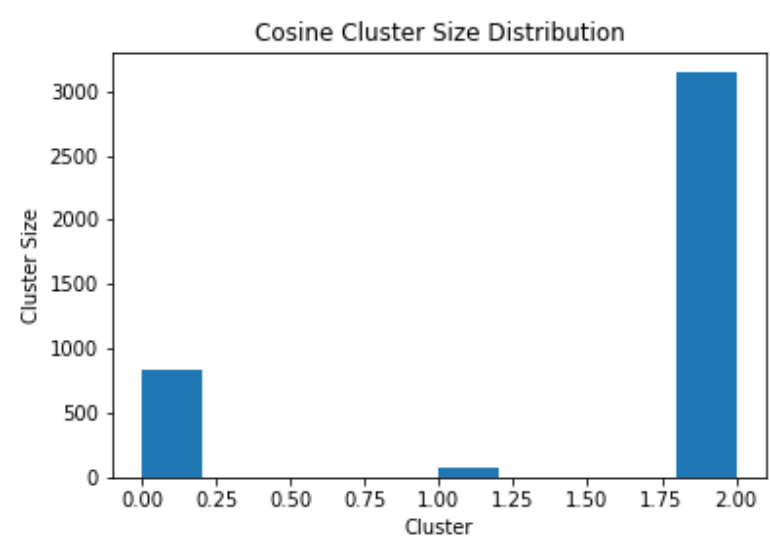
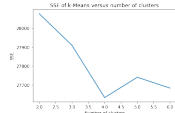


Figure 6. Size distribution of Agglomerative(Cosine)

SSE, BSS and Silhouette scores of each method are listed in Table 2.

/	SSE	BSS	Silhouette score
K-means(Euclidean)	27653.34978	<div><i>davies_bouldin_score: 4.310612</i></div> <div></div>	0.0245
K-means(Cosine)	3919.5702	<i>davies_bouldin_score: 11.72743</i>	0.0014
<i>dbscan (Euclidean)</i>	<i>calinski_harabaz_score: 5784.921361336173</i>	<i>davies_bouldin_score: 1.0572532613361736</i>	0.69

<i>dbscan (Cosine)</i>	<i>calinski_harabaz_score: 4173.402416870325</i>	<i>davies_bouldin_score: 1.0820755354341367</i>	0.72
Agglomerative(Euclidean)	13142168578	111413246643	0.4456
Agglomerative(Cosine)	179224220	240581030	0.0538

Table 2. SSE, BSS and Silhouette score of each method

cluster 1	cluster 2
cn brittany alcoholics alcoholism agonizing affect advice assistant coffee bariatric	dress cookie drinks braingames domestic cn cheetos coffee alcoholics agonizing

Table 3. Sample of Agglomerative Clustering Results

cluster 1	cluster 2
disorder health week lost	survivor robot brain heal

infections	training
affect	apology
reveal	eating
assistant	genes
weightloss	stress

Table 4. Sample of dbscanClustering Results

1.3.2 Visualization

The heatmap of Agglomerative clustering is drawn using seaborn.

The PCA of dbscan clustering is constructed by calling sklearn PCA API, and the results are shown below.

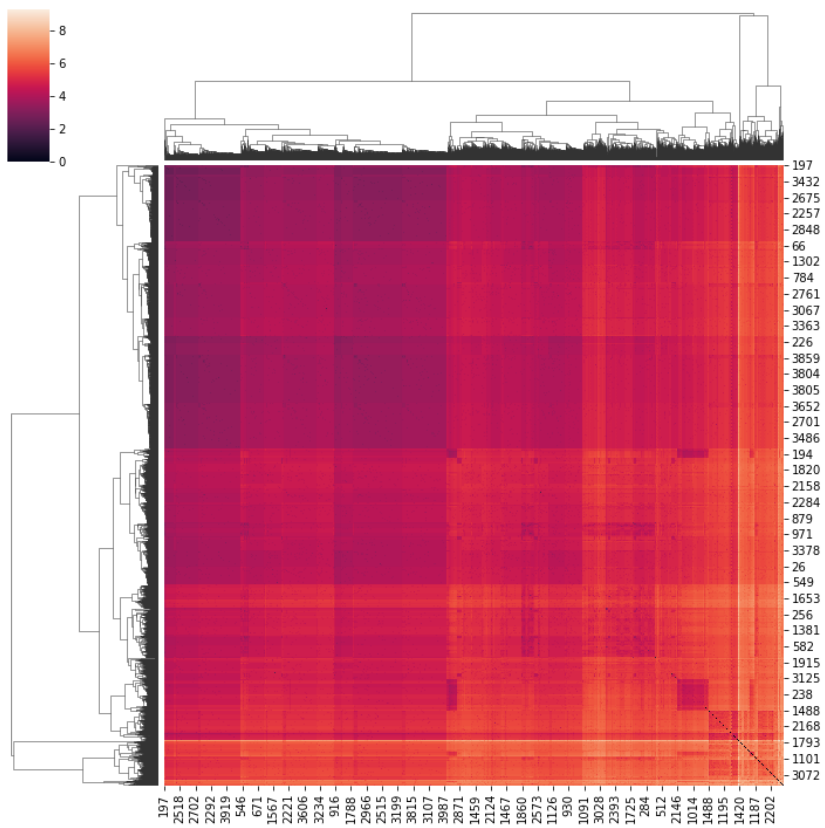


Figure 7. Heatmap of Agglomerative(Euclidean)

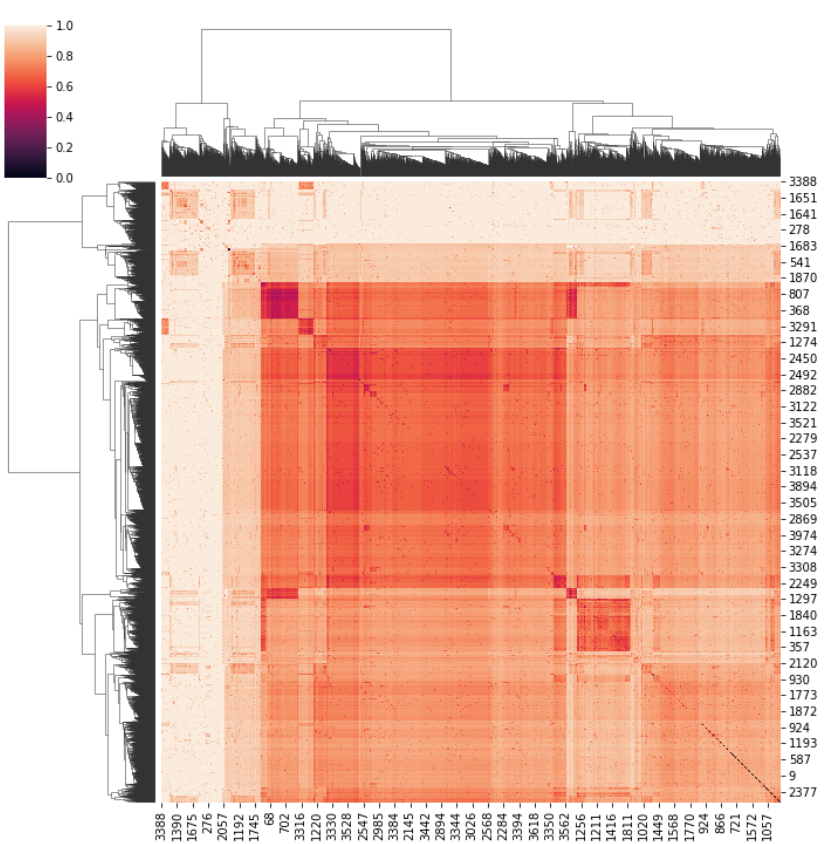


Figure 8. Heatmap of Agglomerative(Cosine)

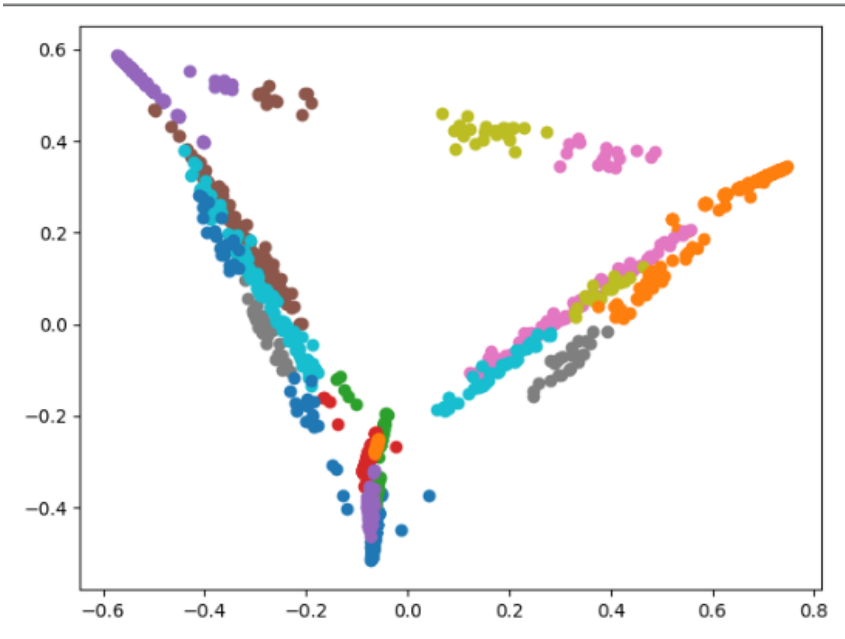


Figure 9. PCA of dbscan(Cosine)

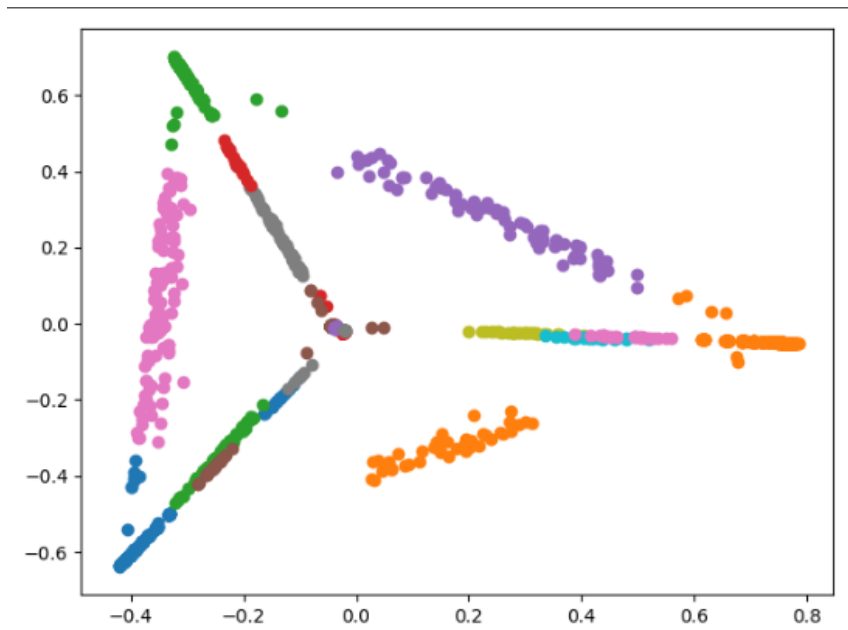


Figure 10. PCA of dbscan(Euclidean)

1.4 Participation

Listed below is the participation of each member.

- Yanbo Huang (dbscan clustering) - 33.3%
- Sinan He(K-means clustering)- 33.3%
- Yunzhou Cao (Agglomerative clustering)- 33.3%

Each team member programmed one clustering method and contributed to the related part of the report. All team members agree with the specified effort.

1.5 Conclusion

To conclude, our team successfully clustered the tweets document using K-means, DBSCAN and Agglomerative clustering. According to the results, we find that Cosine distance measure works better than Euclidean in this case. However, the differences between clustering algorithms are not obvious.

REFERENCES

[1] 2020. *Scikit-Learn*. Accessed March 21. <https://scikit-learn.org/stable/index.html>.
[2] “Clustering text documents using k-means.” 2020. *Scikit-Learn*. Accessed March 21. <https://scikit-learn.org/stable/modules/svm.html>.
[3] “2.3.2 K-means.” 2020. *Scikit-Learn*. Accessed February 14. <https://scikit-learn.org/stable/modules/neighbors.html>
[4] “2.3.7 DBSCAN.” 2020. *Scikit-Learn*. Accessed February 14. https://scikit-learn.org/stable/modules/neural_networks_supervised.html.
[5] “2.3.6 Hierarchical Clustering.” 2020. *Scikit-Learn*. Accessed February 14. <https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees>.
[6] “2.3.10. Clustering performance evaluation.” 2020. *Scikit-Learn*. Accessed February 14. <https://scikit-learn.org/stable/modules/tree.html>

