PR3: Text Clustering

Student name: Yang Chen 013009243

Program Title: Text Clustering using k-means DBSCAN

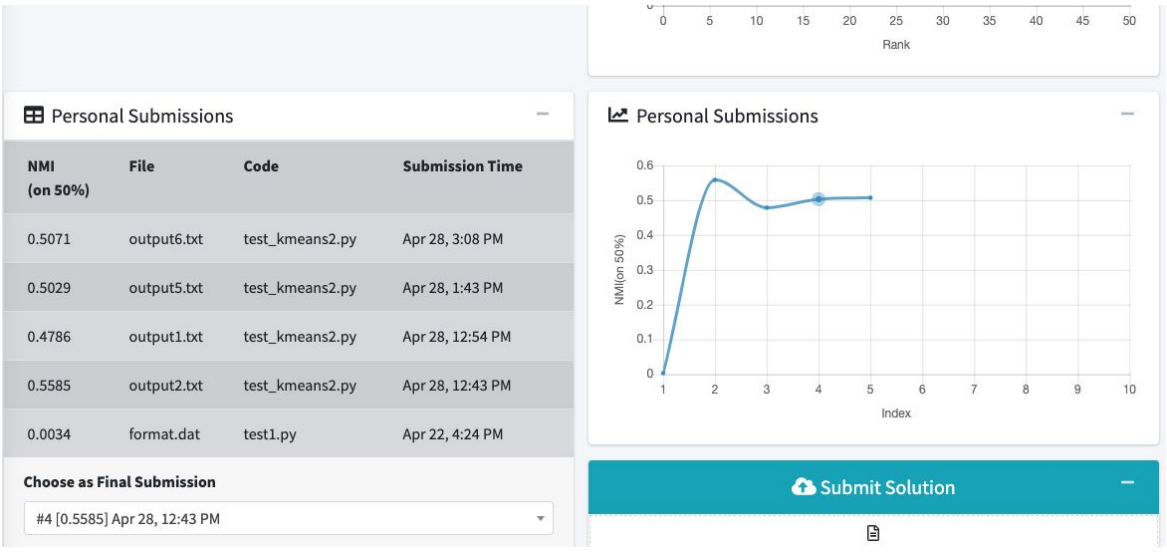
Rank & F1-score: 2 & 0.5585

Program Description: In this program we are going to use the knowledge we learned in class and researched from internet to do text clustering. The Sample file showed about 7 clusters for 8580 row of sparse matrix. The result can be ranked on CLP website. It will provide the rank and NMI score based on the ranking of submissions of different students and the correctness of the answer.

Purposes: Solving the problem of Text Clustering.

Limitations and Findings: This program used NMI to test the accuracy of the program instead of F1. I will be limited to 5 times per day if i can't implement NMI locally.

Found a bug in CLP website. I am trying to select the highest submission 0.5585 during 2nd submit. After i select the 4th one which is reverse order of second. The graph showed me selected a different one. It is quite confusing if i have selected the right one for submission.



Use and Existing clustering algorithm (K-means): I found an simple code sample online to parse a 7 sentences document to vectors and then kmeans from sklearn. It

able to output center points, labels, closest cluster center distance, number of iteration run.

I also read about the library document to generate the data i need.

fit (*X, y=None, sample_weight=None*) [source]

Compute k-means clustering.

Parameters:

X : ***array-like or sparse matrix, shape=(n_samples, n_features)***

Training instances to cluster. It must be noted that the data will be converted to C ordering, which will cause a memory copy if the given data is not C-contiguous.

y : ***Ignored***

not used, present here for API consistency by convention.

sample_weight : ***array-like, shape (n_samples,), optional***

The weights for each observation in X. If None, all observations are assigned equal weight (default: None)

Attributes:

cluster_centers_ : ***array, [n_clusters, n_features]***

Coordinates of cluster centers. If the algorithm stops before fully converging (see `tol` and `max_iter`), these will not be consistent with `labels_`.

labels_ :

Labels of each point

inertia_ : ***float***

Sum of squared distances of samples to their closest cluster center.

n_iter_ : ***int***

Number of iterations run.

```
vectorizer = TfidfVectorizer(stop_words='english')
print(vectorizer)
X = vectorizer.fit_transform(documents)
#print (X)


lines = vectorizer.fit_transform(lines)
print(lines)


true_k = 100
model = KMeans(n_clusters=true_k, init='k-means++', max_iter=100, n_init=1)
#model = KMeans(n_clusters=2, random_state=0)
model.fit(lines)
```

Implement a variation of the DBSCAN clustering algorithm: First i get values from matrix about the average distance then observe the distance to find a possible radius and eps. Then i will write iterations to try different configuration and let it run.

```
eps = 5
radius = 0.5
minDis = 99
minDisList = []
for i in range(len(model.cluster_centers_)):
    minDis = 99
    for j in range(len(model.cluster_centers_)):
        cosDistanceTemp = cosDistance(model.cluster_centers_,i,j)
        print("cosDistanceTemp ",cosDistanceTemp )
        if cosDistanceTemp[0] > minDis:
            minDis = cosDistanceTemp[0]
    minDisList.append(minDis)
print("minDisList ", minDisList)
```

```
#update results based on eps and radius
for i in range(len(clusters)):
    count = 0
    for j in range(len(clusters)):
        if cosDistance(model.cluster_centers_,i,j) < radius:
            results[j] = results[i]
```