IR CSCE 670 HW3 Report

Part 1 : K Means Clustering

I have saved each query result as a new text file and name them ranging from 1.txt to 150.txt saved in the folder called docs.

1.txt – 30.txt : texas aggies
31.txt- 60.txt : texas longhorns
61.txt - 90.txt : duke blue devils
91.txt – 120.txt : dallas cowboys
121.txt – 150.txt : dallas mavericks

I am querying using request0. Which can be imported from request-transitions.
I have handled the duplicate results but only when the URL is same, but if the URL are different and Title is same, I am not assuming it as a duplicate result. I am also querying till I get 30 results for each query by using $skip method in the request.

In this part of the assignment we have implemented a k-means algorithm. K means is about selecting random K points from the data points and treat them initial centroid. So accuracy of the clustering is highly dependent on the initial centroid chosen. For each K, we ran over algorithm and if the RSS value converges to certain threshold, we stop the re calculation of centroid and re-assignment of the centroid points. I am using Cosine similarity between the documents for assigning the document to a particular cluster. I am calculating cosine similarity of a document to all the existing centroids and assign that doc to that centroid with maximum cosine value.

I have run my algorithm for different values of K, and my best result is coming out for range of values from 6-9. I have fixed the value of K=8, in the code and for this value also, value of purity and Rand indexing is varying as it depends on how initial points got selected. Below are the graphs attached for the different values of RSS vs K, Purity Vs K and Rand Index vs K.

GRAPH:

RSS vs K:

RSS on Y-axis and K on X-axis
From the graph we can observe that there is knee point at K = 8 (Curver Flattens here) So we have chooses the value of K for our calculation as 9 and fixed that value in code for testing purpose.

Purity Calculation:

For each cluster we have predicted from Kmeans algorithm, we have created five lists for each of the query category and assign each document in that cluster to that category list. In that way we have calculated number of documents of each actual class in our predicted cluster. We than take the maximum value among the lengths of lists for each cluster formed and sum them (ClusterSum).

Then Purity of the clustering can be calculated by:

Purity = ClusterSum/total number of documents

Here for K =9: We have got maximum value for length of each cluster as 11,15,20,8,12,16,6,7,6.

So ClusterSum = 101

Total Docs: 150

Purity: 101/150 = 0.67333

Rand Index Calculation:

We have created a pair of each document with other giving us a total of 150C2 pairs. We already knew by the queries results about the true class of the results. With these documents we cluster them based on the K means algorithm and divide them into K clusters.

Now for each pair we already knew about their true class which they belong by the document numbers and we also predicted their new cluster based on our kmeans algorithm.

Based on 2x2 contingency table as shown below we calculated values for each cell in the matrix depends on their actual class and predicted clusters.

	Same cluster	**Different cluster**
Same class	True positive	False negative
Different class	False positive	True negative

Once the values of TP, FP, FN and TN got calculated we calculated value of Purity by this formula:

Rand Index = (TP+TN)/(TP+FP+FN+TN)

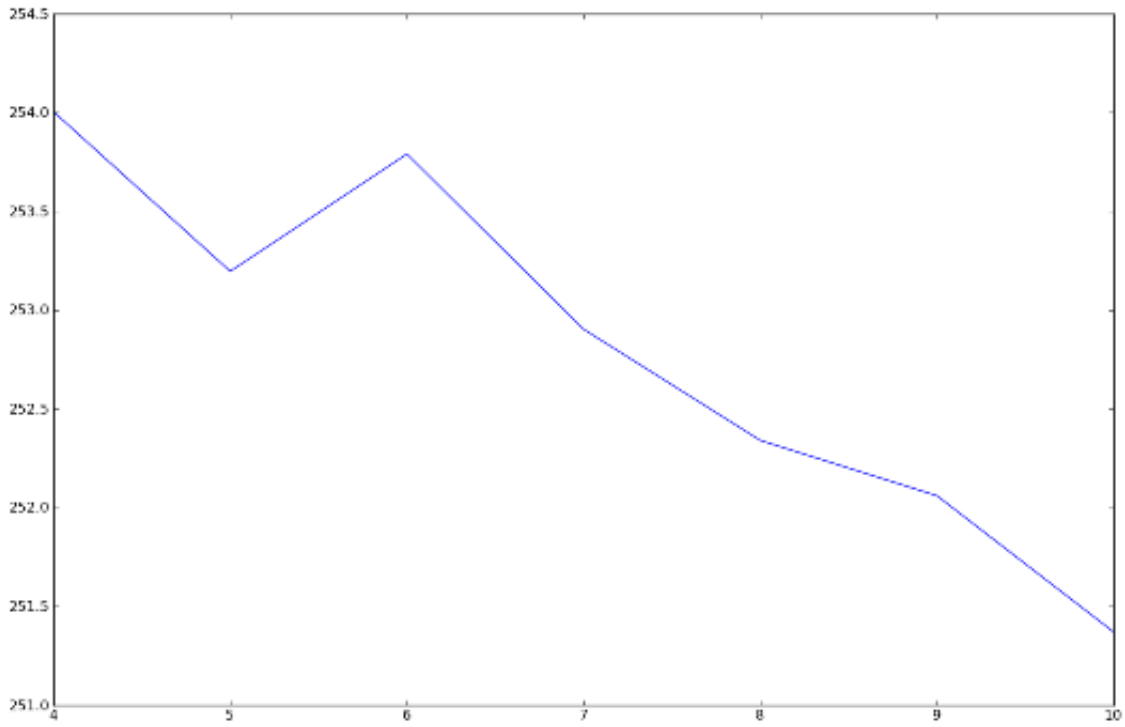
For K=9:

We have got our results converged with RSS Value : 253.102 after 4 iterations.

We have got TP, TN, FP, FN: 1517, 16388, 1372, 2924 respectively.

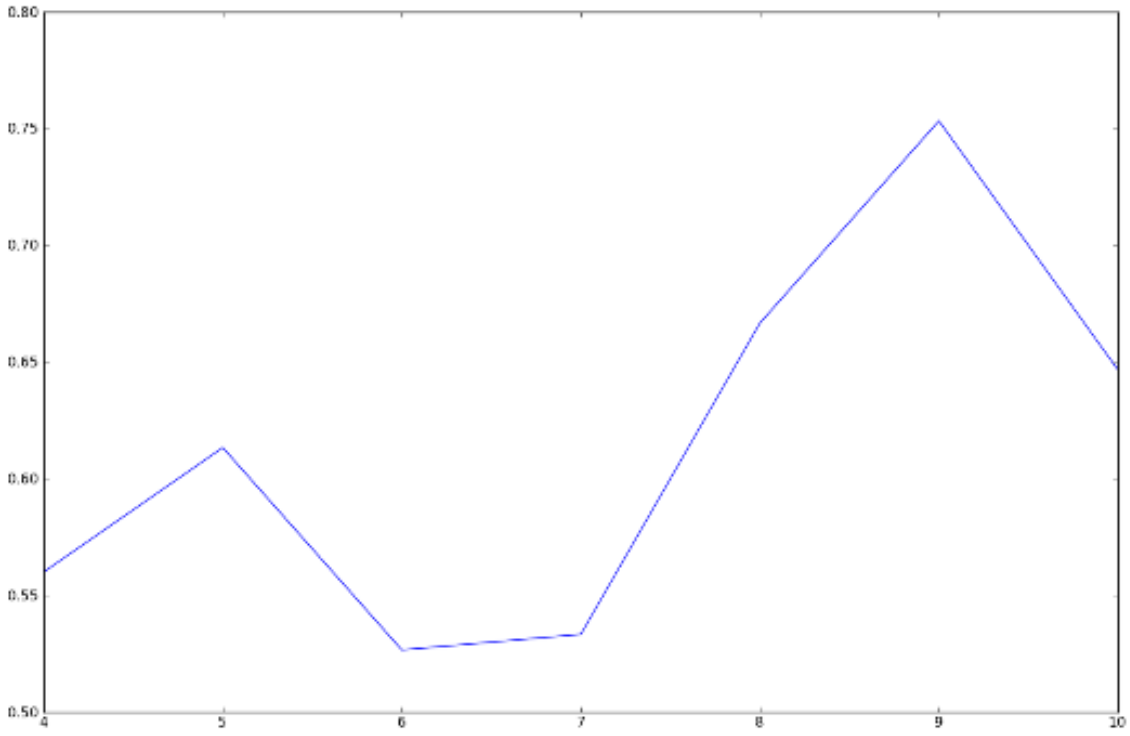
RandIndex : 0.806495202919

RSS vs K



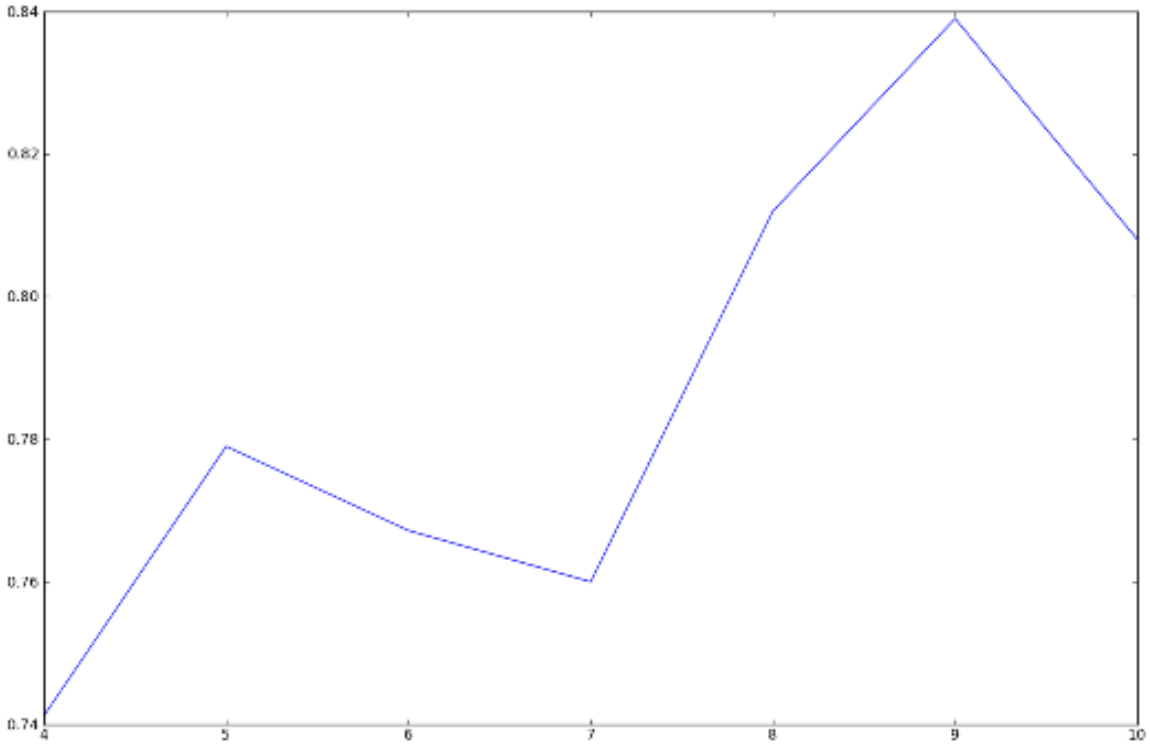
PURITY vs K

Value of the purity is highest at the K = 9, value we have selected from our calculation and after that value of the purity decreases sharply making choice of our K is optimal.



RAND INDEX vs K

Value of the Rand Index is highest at the K = 9, value we have selected from our calculation and after that value of the Rand Index decreases sharply making choice of our K is optimal.



Part 2: Naïve Bayes Classifier

In this part of the homework, we have implemented the Naïve bayes classifier for the 15 queries for 3 categories and 30 results for each category, So in total we have 1350 documents. Now we run our naïve bayes classifier algorithm and it classifies our documents into 3 categories. Since we already knew that which document belongs to which category actually and what will be their predicted class. From these value we calculate tp,fp,fn and tn for all the classes together and with that calculate precision and recall.

So we have 3 categories as:

1.txt – 450.txt : Entertainment

451.txt – 900.txt: Business

901.txt - 1350.txt: Politics

Now we calculate the probability of each term in each class and created our training dataset.

Now when any test document comes, we assign them a category based on the naïve bayes algorithm. Now we know for each document its actual class based on the query results and its predicted class based on the algorithm.

So for each document in all the categories we calculate the values of Confusion matrix, tp,fp,fn and tn in the following manner:

if document actual class and predicted class are same**: increase TP**

if document actual class is different but predicted in different class: **Increase FP**

if document is not predicted to in the class but its actually in that classs: **Increase FN**

if documents is not predicted to in that class and its not in that class : **Increase TN.**

Confusion Matrix

	In the class	**Not in the class**
Predicted to be in the class	True positive	False positive
Predicted to not be in class	False negative	True negative

We calculated these values for each document of all the categories and take the cumulative sum of them to calculate F value. Each Document can satisfy True positives once but can satisfy false positive and false negative more than once. We also got the false positive and false negative same. Thus precision and recall values are supposed to be same and in turn MicroAvg F.

P = TP / (TP + FP)

R = TP / (TP + FN)

F = 2PR / (P + R)

So for Example we run our code on the sample Test Queries. We have 150 documents for each 3 categories. Results obtained are as:

tp,fp,fn,tn : 330.0 111.0 111.0 441.0

Precison and Recall are : 0.748299319728 0.748299319728

MicroAvg F is : 0.748299319728

Part 3: Improvement in Clustering

The basic limitation of Kmeans algorithm is the random selection of k starting points from the data points given which we need to divide into different clusters. So if the random point's choosen are very similar then results would be very different and similarly if choosen very different then result obtained might be better. So in order to avoid the randomization we can have choose the initial points from some algorithm such that initial centroid points we select must belongs to different clusters and should have least similarity. We have two ways of selecting the initial centroid. I am discussing here both the approaches and also implemented both of them, but I found first approach discussed here more effective and giving me better results, so I kept that in code running and comment out the second method of choosing initial point.

- **Apart from the selection of initial centroid points, I have given extra weightage to the title of the document. I have given twice weightage to the terms in the title.**

- I have also given additional weightage to the words which I can extract from the URL. E,g URL – **http::://aggieathletics.com/aggies-beat-alabama-in-sec-championship.html**

  So I have extract the words from the URL of the document which I can get from the JSON object. So I consider the words like aggieathletics,aggies,beat,Alabama,sec, championship while creating Training data set. So if any test document if contains the same words then its probability of going to that particular class increase.

- The final clustering result of the kmeans clustering algorithm greatly depends upon the correctness of the initial centroids, which are selected randomly. The original k-means algorithm converges to local minimum, not the global optimum.

  In both the below methods a new approach is proposed by the authors for finding the better initial centroids and to provide an efficient way of assigning the data points to suitable clusters with reduced time complexity.

Method 1:

Reference : *farthest-point heuristic based initialization methods for k-modes clustering*
Zengyou He,Department of Computer Science and Engineering, HIT China

In this paper author describes the algorithm. Pick a random arbitrary point s1. Pick a point s2, which Is maximum distance from s1. Now we have two centroid which are farthest from each other. Now pick Si to maximize the distance to the nearest of all centroids picked so far. That is maximize the min (dist (s1,si),dist(s2,si)…….)). After all K representatives are chosen we can define our initial dataset as of these points. This algorithm ensures that all the points selected must be least similar to each other.

For implementation of this algorithm, I have calculated distance of each point from other. I have sort the values of each document distance from the origin and picked s1 to be the first document and s2 to be the second document so that they are maximum apart from each other. For calculating further points I have calculated the minimum distance of each point from these centroid and save them in a list and among these centroids I find out the point which has a maximum value, which ensures that data point we have selected is not near to any of the existing centroids.

After implementation of this method I have not got so improved results of Purity, randindex but this should give better results on the varying training data set. For K = 8, I have got knee in the curve for graph RSS vs K . So I fixed value to be k = 8 in the code. Values of Purity obtained at this K are 0.6866 and Randindex to 0.824.

GRAPH:

RSS vs K:

RSS on Y-axis and K on X-axis
From the graph we can observe that there is knee point at K = 7 (Curver Flattens here) So we have chooses the value of K for our calculation as 7 and fixed that value in code for testing purpose.

Purity Calculation:

For each cluster we have predicted from Kmeans algorithm, we have created five lists for each of the query category and assign each document in that cluster to that category list. In that way we

have calculated number of documents of each actual class in our predicted cluster. We than take the maximum value among the lengths of lists for each cluster formed and sum them (ClusterSum).

Then Purity of the clustering can be calculated by:

Purity = ClusterSum/total number of documents

Here for K =7: We have got maximum value for length of each cluster as 20,21,8,23,8,14,9

So ClusterSum = 103

Total Docs: 150

Purity: 101/150 = 0.6866

Rand Index Calculation:

We have created a pair of each document with other giving us a total of 150C2 pairs. We already knew by the queries results about the true class of the results. With these documents we cluster them based on the K means algorithm and divide them into K clusters.

Now for each pair we already knew about their true class which they belong by the document numbers and we also predicted their new cluster based on our kmeans algorithm.

Based on 2x2 contingency table as shown below we calculated values for each cell in the matrix depends on their actual class and predicted clusters.

	Same cluster	**Different cluster**
Same class	True positive	False negative
Different class	False positive	True negative

Once the values of TP, FP, FN and TN got calculated we calculated value of Purity by this formula:

Rand Index = (TP+TN)/(TP+FP+FN+TN)

For K=7:

We have got our results converged with RSS Value : 248.551 after 4 iterations.

We have got TP, TN, FP, FN: 1864, 16308, 1452, 2428 respectively.

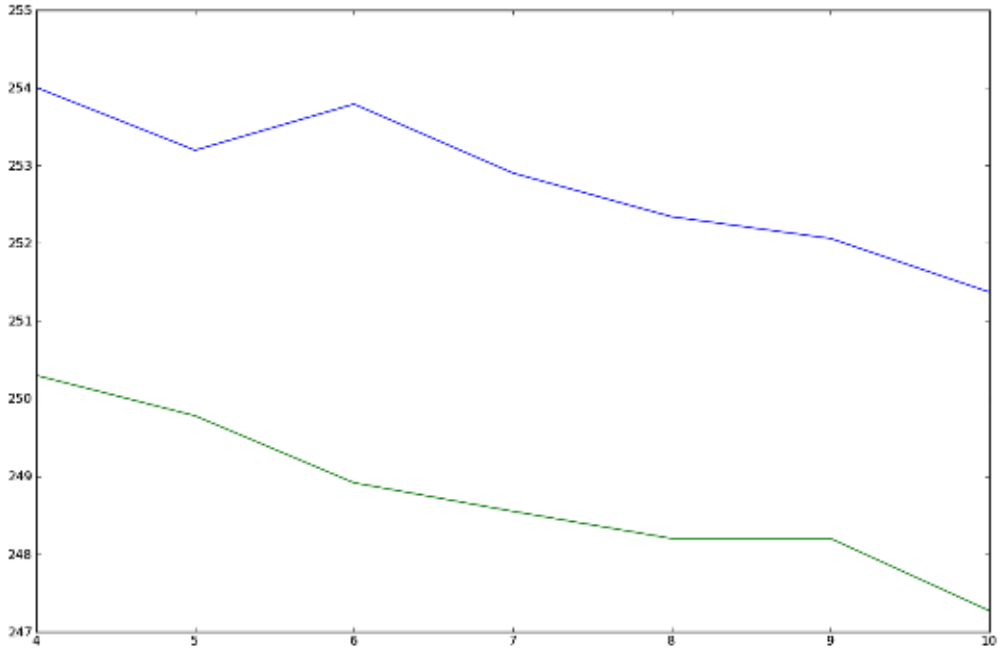
RandIndex : 0.82405

RSS vs K

K on X-axis and RSS on Y-Axis

Blue Line: Part 2 Basic Clustering

Green Line: Part 3 Clustering with Improvements

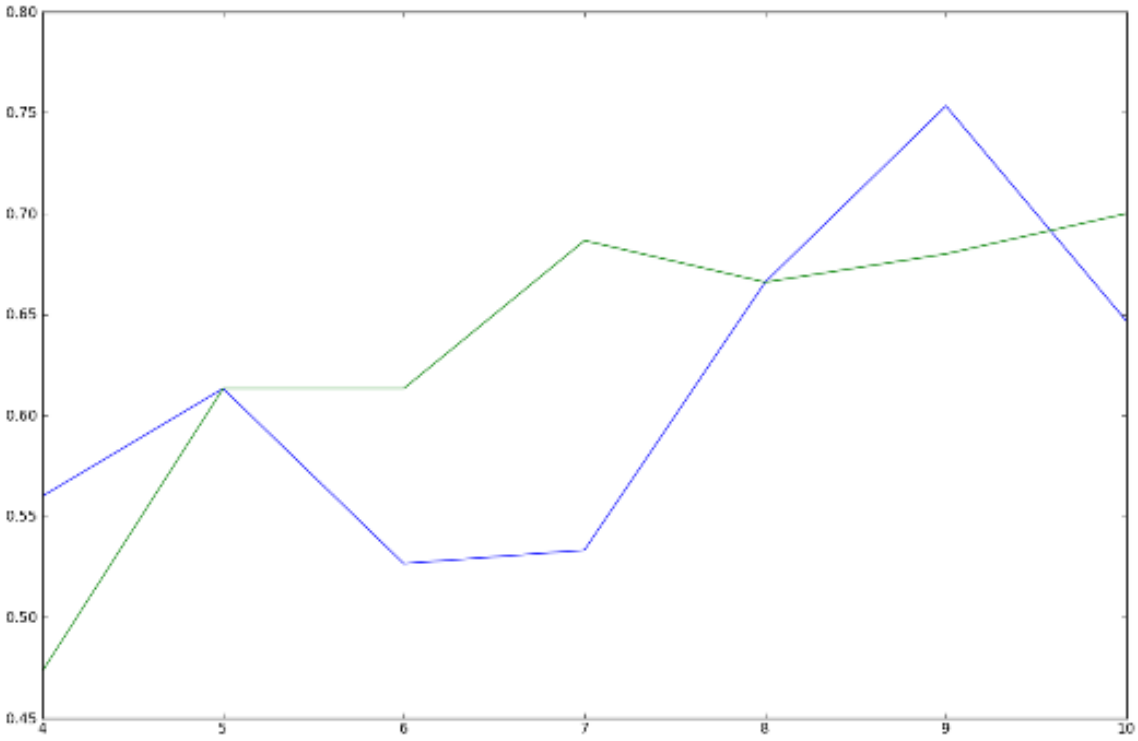


Purity Vs K

K on X-axis and Purity on Y-Axis

Blue Line: Part 2 Basic Clustering

Green Line: Part 3 Clustering with Improvements

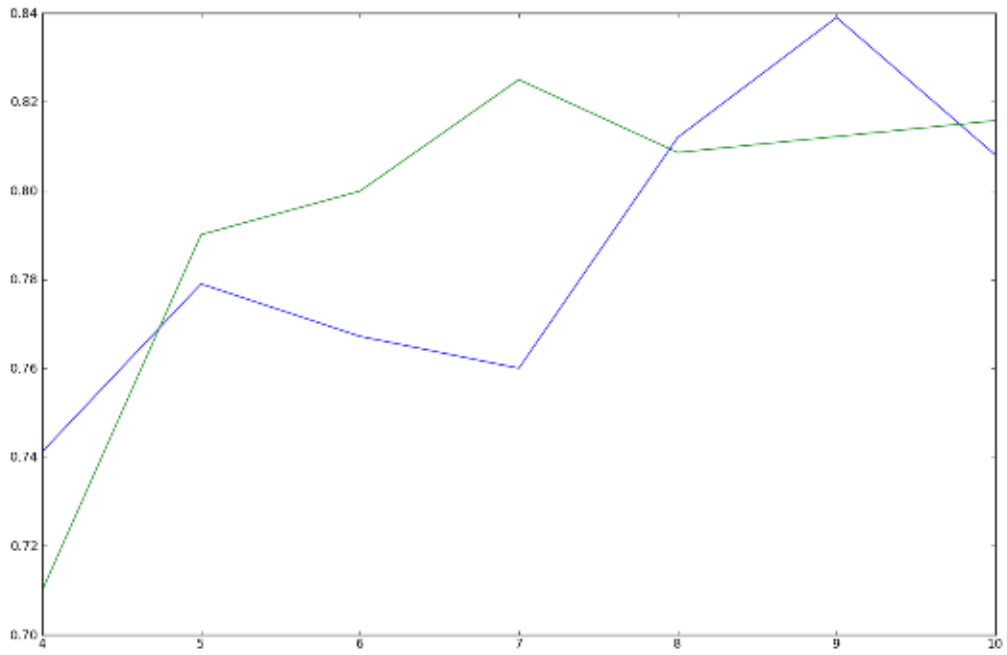


Rand Index Vs K

K on X-axis and RandIndex on Y-Axis.

Blue Line: Part 2 Basic Clustering

Green Line: Part 3 Clustering with Improvements



Method 2:

Reference: *Enhancing K-means Clustering Algorithm with Improved Initial Center*
Madhu Yedla#1, Srinivasa Rao Pathakota#2, T M Srinivasa

I have implemented the algorithms described below. In this algorithm, for each data point we calculate the distance from origin. Then, the original data points are sorted accordance with the sorted distances. After sorting partition the sorted data points into k equal sets. In each set take the middle points as the initial centroids. These initial centroids lead to the better unique clustering results. Next, for each data point the distance calculated from all the initial centroids. The next stage is an iterative process which makes use of a heuristic approach to reduce the required computational time. The data points are assigned to the clusters having the closest centroids in the next step the present nearest distance from closest centroid as in the original k-means algorithm.

But this algorithm has limitation as some points that are equidistance from origin but in different direction can be clubbed in same cluster. So it's optimal for cases when points are mostly directed in the same direction. The results improved by this method are negligible.

Code for this section is commented our in the part3_cluster.py at line number # 381

Part 3: Improvement in Classification

For Improvement in the naïve bayes classification we have used the concept of feature selection as taught in the class. Since not all the terms in the class correctly specifies the class. Many words are rare and they mislead the classifier. So eliminating them from the training data set improves the efficiency and effectiveness of the text classification.

So for improving the efficiency of the classifier I am here implementing the **Mutual Information** based Feature selection, in which only top words which contributed for the class are calculated and rest of the terms can be neglected from the training dataset.

So Mutual Information can be calculated by finding the four values N00, N11, N10 and N01 for each feature in each class. And from them finds the MI value for them. Top few thousand words can be considered for training set based on MI value and rest of them can be ignored.

For each Feature for each Class Calculate:

N11: Number of documents that contain feature and are in class.

N10: Number of documents that contain feature and are not in class.

N01: Number of documents that do not contain feature and are in class.

N00: Number of documents that do not contain feature and are not in class.

N = N00 + N11 + N01 + N10

MI = N11/N * math.log(((N*N11)/(N11+N10)* (N01 + N11)),2) + N01/N * math.log((N*N01)/((N00 + N01) * (N11 + N01)),2) + N10/N * math.log((N*N10)/((N10 + N11) * (N10 + N00)),2) + N00/N * math.log((N*N00)/((N10 + N00) * (N01 + N00)),2)

So after removing extra features, we can classify the each test document based on the terms in the selected feature list only. As we have find the values of tp,fn,fp and tn for each class in the part 2, we also finds the values of them here and find out the increase of 2% value in Microavergae F value when I included top 4000 features in each class. As compared to part 1 in which 330 out of 450 documents was correctly classified here I am getting little increase with 336 documets correctly classified.

tp,fp,fn,tn : 336.0 105.0 105.0 441.0

Precison and Recall are : 0.761904761905 0.761904761905

MicroAvg F is : 0.761904761905

- **Apart from the selection of feature selection, I have given extra weightage to the title of the document. I have given twice weightage to the terms in the title.**

- I have also given additional weightage to the words which I can extract from the URL. E,g URL – **http::://aggieathletics.com/aggies-beat-alabama-in-sec-championship.html**

  So I have extract the words from the URL of the document which I can get from the JSON object. So I consider the words like aggieathletics,aggies,beat,Alabama,sec, championship while creating Training data set. So if any test document if contains the same words then its probability of going to that particular class increase.

- Also Tried removing stop words from the corpus but it will decrease the value of F, so I again consider them in my result.