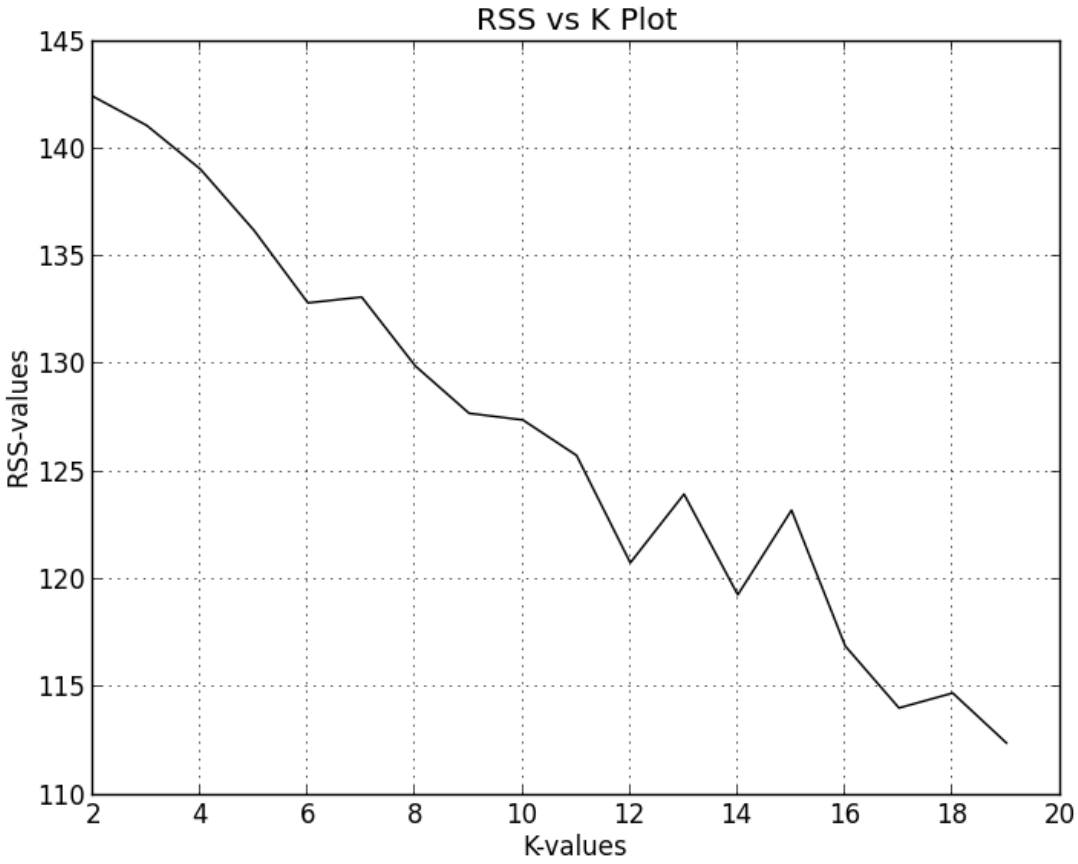
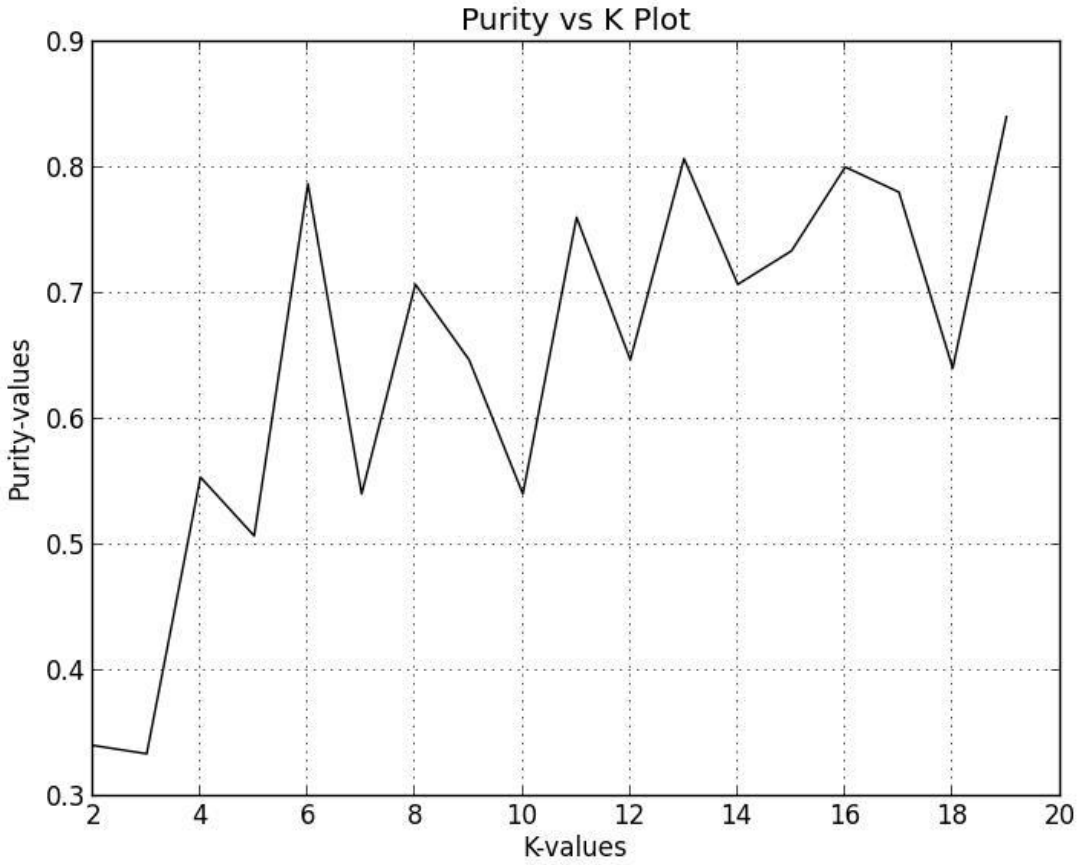
Report for IR-Assignment-3
Submitted by: Sanjeev Kumar Singh (UIN: 922002623)

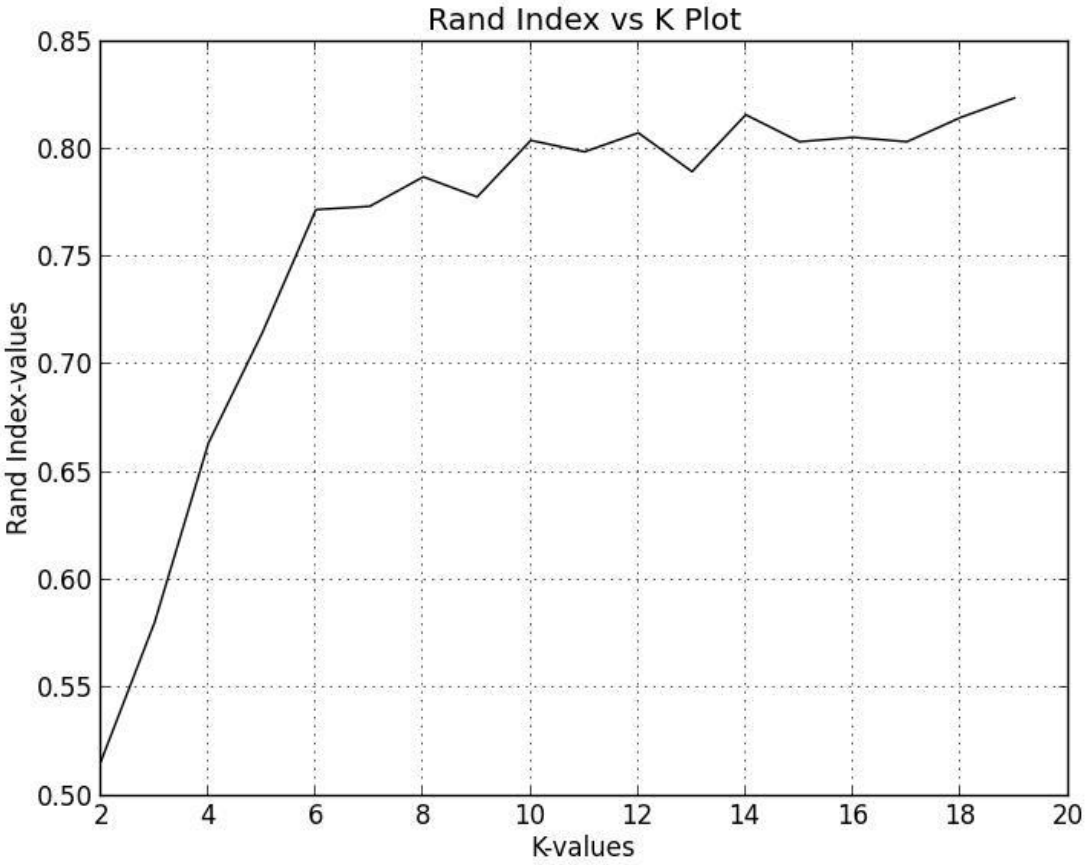
Part1:



Yes, we can observer a few knees in the curve. Example - as we can observe for K=6 the value of RSS is approximately 132 and even for K=7, the RSS value is almost the same.



As we can observe from the above graph, purity initially increases and reaches to a peak at K=6. We have already observed a knee in the RSS Vs K graph for K=6. However, the overall graph of Purity Vs K appears to be depicting an increment in Purity with the increase in K. On the contrary, in RSS Vs K graph, overall the value of RSS seems to be decreasing as the value of K increases. Thus both these graphs are showing in-lined behavior in some sense as low RSS means that points of a cluster are very close to the centroid, which in other sense will lead to an increment in the purity.



We can easily observe from the graph that till K=6, the value of Rand Index is increasing at a very fast pace and after K=6, the overall value of Rand Index appears to be increasing but at a slower rate. Again we can observer a knee in the graph between K=6 and K=7. Thus again the behavior of Rand-index Vs K graph and Purity Vs K graph appears to be inlined.

Thus, as per my observation all the three evaluation criteria are in sync with each other.

Calculation of Purity and Rand Index for K=6:

Step 1: Queried on Bing with the given queries and tried retrieving at most 30 results for each query, thus got my repository to work on. In this case, “repository_part1.txt” is the repository. Each line of this file contains description + title for each of the queried result. Thus 150 lines of “repository.txt” represent the retrieved description + title for different queried results.

Note: I am considering each description + title as a document.

Step 2: tokenized each of these documents and stored in a list. Thus, got a list containing lists of tokenized documents.

Step 3: using these tokenized lists, created dictionary with key as term/token and value as corresponding TF value.

Step 4: updated the dictionary with idf value and then normalized it and finally got a dictionary with key as docID and value as dictionary with key as term/token and value as normalized TF-IDF value.

Note: For me docID is simply the line number of the repository corresponding to a particular doc.

Step 5: Also created a vector containing all the unique terms in the entire corpus and sorted it. This is going to be my blueprint for representing any document as a vector in n-dimensional space, where each of these unique terms will act as a dimension in itself.

For my case there are 1670 unique terms, so I am playing in 1670 different dimensions.

Step 6: followed the standard algorithm of K-means for clustering i.e choosing K(say 5) random seeds as the starting points and then clustering the rest of the documents based on their Euclidian distance from these seeds. Then recalculated the centroids for each of these clusters and kept doing this till the time convergence condition is met. If for 5 consecutive times , I get the same cluster or there is no change in the cluster, then I consider it as convergence.

Once the clustering is done properly, I save the results in a dictionary with key as the cluster name and value as a list which contains the docIDs belonging to that particular cluster.

cluster_4 [8, 15, 22, 31, 33, 34, 38, 55, 112]

cluster_5 [24, 42, 50, 53, 62, 64, 65, 66, 67, 69, 70, 71, 73, 74, 76, 77, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 89]

cluster_2 [6, 29, 59, 95, 96, 98, 120, 121, 122, 131, 132, 134, 139, 140, 142, 143, 149]

cluster_3 [0, 1, 3, 4, 5, 7, 9, 10, 11, 16, 17, 18, 25, 26, 27, 28, 30, 32, 35, 36, 37, 39, 40, 43, 44, 45, 46, 47, 48, 49, 51, 52, 57, 58, 60, 61, 63, 72, 80, 91, 93, 99, 110, 114, 117, 118, 130, 138]

cluster_0 [2, 12, 13, 14, 19, 20, 21, 23, 41, 54, 56, 68, 75, 90, 92, 94, 97, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 111, 113, 115, 116, 119, 124]

cluster_1 [123, 125, 126, 127, 128, 129, 133, 135, 136, 137, 141, 144, 145, 146, 147, 148]

Note: Since I queried each of the query terms sequentially and stored them in 'repository.txt' sequentially too. So, just the docID is enough for me to know the class to which that particular doc belongs to. Example – doc with ID 24 and 17 belongs to the same class as 24/30 = 0 and 17/30 = 0 (considering the integral part after division). However, doc with IDs 70 and 117 belong to different classes as 70/30 = 2 and 117/30 = 3.

Step 7: Calculation of purity-
I have my cluster ready and I am aware of the classes of each of the docs belonging to those clusters. Moreover, I also know the total number of docs (N) in entire corpus. Now let's calculate purity.

I calculated the number of docs from different classes belonging to that particular cluster stored that in a list. Then took the maximum value existing in that list. I did the same for each of the clusters and then summed up all the max values obtained.
Finally divided the sum by the total number of the docs in the entire corpus thus got purity for my clustering.

cluster_4 [3, 5, 0, 1, 0]
cluster_5 [1, 3, 23, 0, 0]
cluster_2 [2, 1, 0, 3, 11]
cluster_3 [16, 18, 5, 7, 2]
cluster_0 [8, 3, 2, 19, 1]
cluster_1 [0, 0, 0, 0, 16]

sum of max values = 5 + 23 + 11 + 18 + 19 + 16 = 92
total_no_docs = 150

therefore, purity = sum of max values/ total_no_docs

Step 8: Calculation of Rand Index:-
Total number of document pairs = NC2 = (total no of docs) * (total no of docs -1)/2 = 11175
Moreover, total number of document pairs = TP + FP + FN + TN

TP = number of pairs of documents in the same class and same cluster = 897
Since, I already have number of docs from different classes belonging to different clusters. So, we can calculate the required number of pairs easily.
Example – for cluster_4
TP = 3C2 + 5C2 (C represents combination here)
Similarity calculated TP for each of the clusters and then summed them up to get final TP.
Similarly, calculated pairs of documents by following the standard definition as the pair of documents lying in different classes and different clusters.
In this case TN = 8040

Rand Index = (TP + TN)/(TP + FP + FN + TN)
= (TP + TN)/ total number of document pairs

Thus, purity and Rand Index are calculated.

Part- 2:

After classifying the documents from different test sets, I got following result:
politics_testSet:
politics 111
business 23
entertainment 16

entertainment_testSet
politics 14
business 29
entertainment 107

business_testSet
politics 19
business 119
entertainment 12

Now the calculation of different parameters for calculating F1 is done as:
True Positives (TP) is the number of documents that are predicted to be in the class and are actually in the same class.
Example – For Entertainment test set, there are 107 documents which are classified to be in the entertainment class. So, TP for this test set is 107
Similarly, I calculated the TPs for different test-sets/classes

False Positives (FP) is the number of documents that are predicted to be in a particular class, but they are classified not be in the class.
Example- For Entertainment test set, there are (14 + 29) documents which are classified in the non-entertainment classes. So FP for this test set is 14 + 29 = 43

False Negatives (FN) is the number of documents that are not predicted to be in the class, however they are classified to be in that particular class.
Example – If we consider Entertainment test set, there are 300 documents which are predicted not to be classified as entertainment, however amongst these 300 documents there are a few which are still classified as entertainment. In this case there are 43 such documents.

True Negatives (TN) is the number of documents that are not predicted to be in the class and they are also not classified to be in that class.
For example: If we consider Entertainment test set, there are 300 documents which are predicted not to be classified as entertainment and amongst these 300 documents there are 257 documents which satisfy the criteria. So, for this case TN = 257

Thus I calculated all the above parameters for different classes/test sets and then summed them up to get their aggregate values. Then I calculated Precision and Recall by using these aggregate values using this formula:

Precision(P) = TP/(TP + FP)
Recall(R) = TP/(TP + FN)

Once I calculated Precision and Recall values then calculated Microaveraged F1 by:
Microaveraged F1 = (2*P*R)/ (P+R)

Confusion Matrices:

Entertainment:
TP= 107
FP= 43
FN= 43
TN= 257

Business:
TP= 119
FP= 31
FN= 31
TN= 269

Politics:
TP= 111
FP= 39
FN= 39
TN= 261

Aggregate:
TP= 337
FP= 113
FN= 113
TN= 787

Precision(P) = TP/(TP + FP) = 337/(337 + 113) = 0.74888888888
Recall(R) = TP/(TP + FN) = 337/(337 + 113) = 0.74888888888

Microaveraged F1 = (2*P*R)/ (P+R) =

(2 * 0.74888888888 * 0.74888888888)/ (0.74888888888 + 0.74888888888)
= 0.74888888888

Part-3:

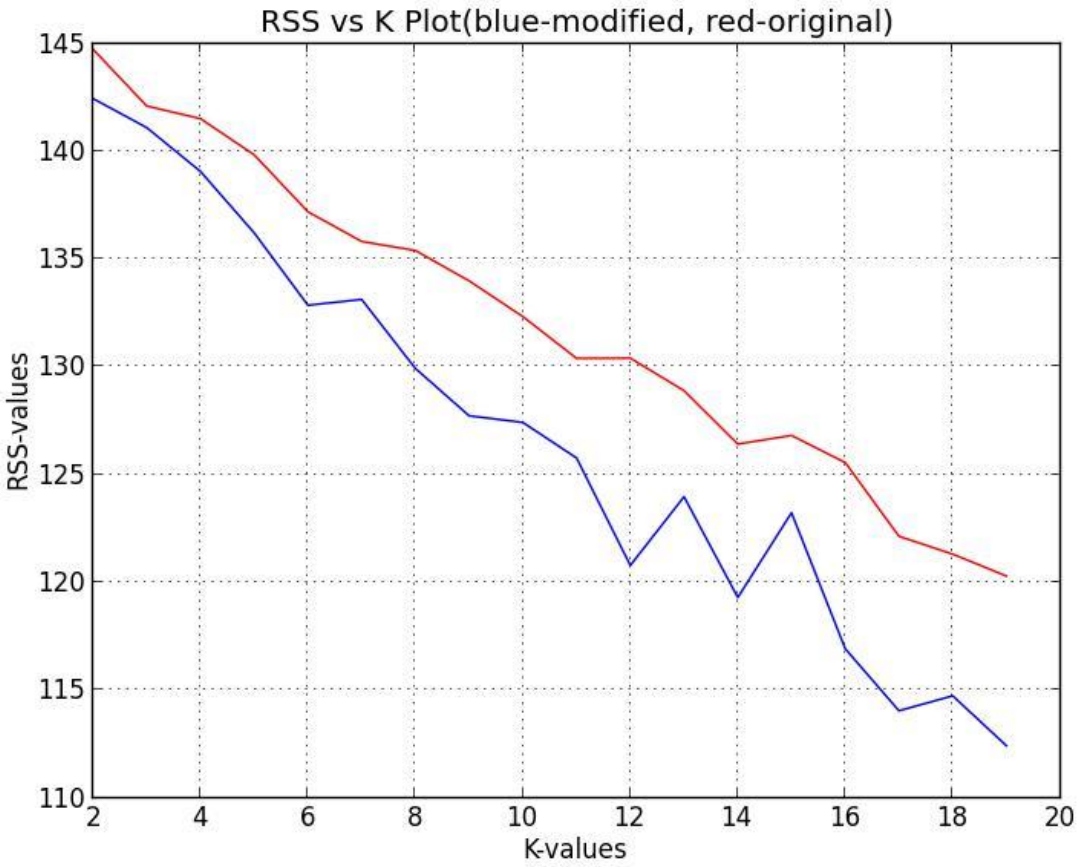
Modified Clustering:

Improvement measures taken for clustering:

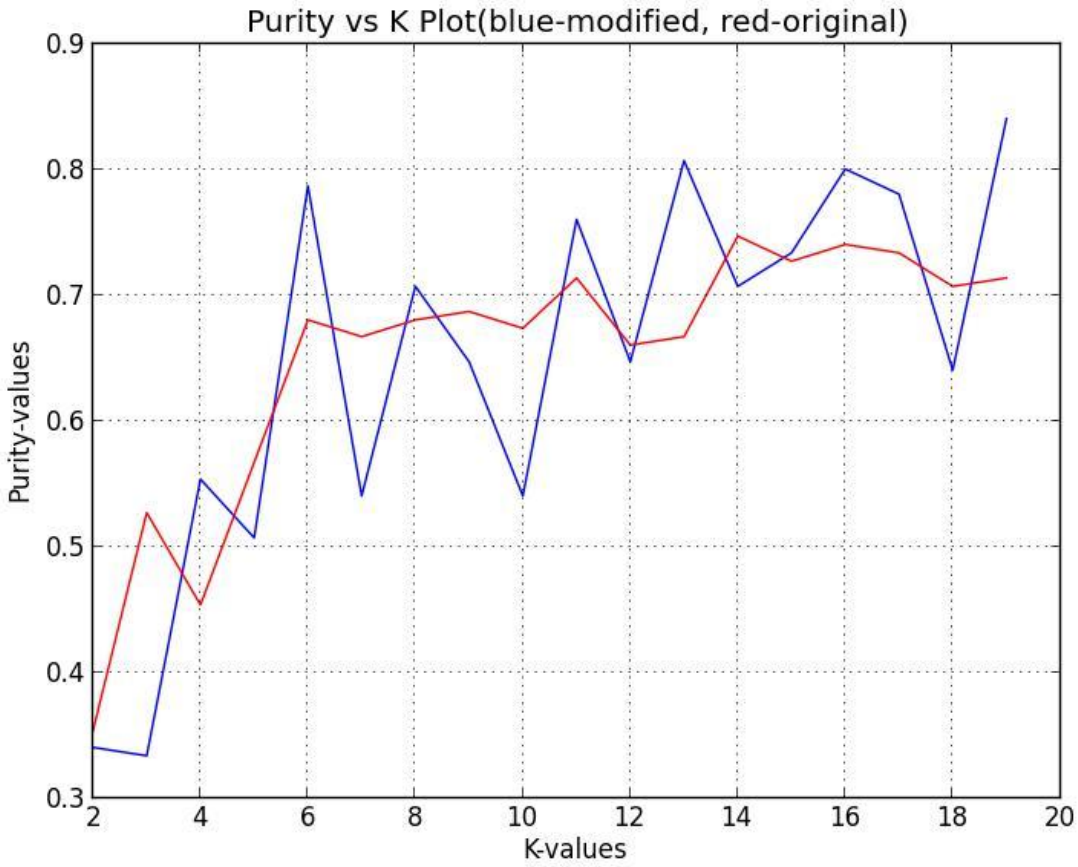
1. Removal of duplicate documents - In part one I just queried for 30 documents for each of the queries, however I didn't check whether any duplicate document exists or not. For the improved version I tried collecting 30 unique documents, which improved clustering drastically. Actually the presence of duplicate documents hinders better clustering.

2. Removal of stop words – Since stop words hardly contribute to the meaningfulness of data. So, removal of stop words again led to some improvement in the modified clustering approach.

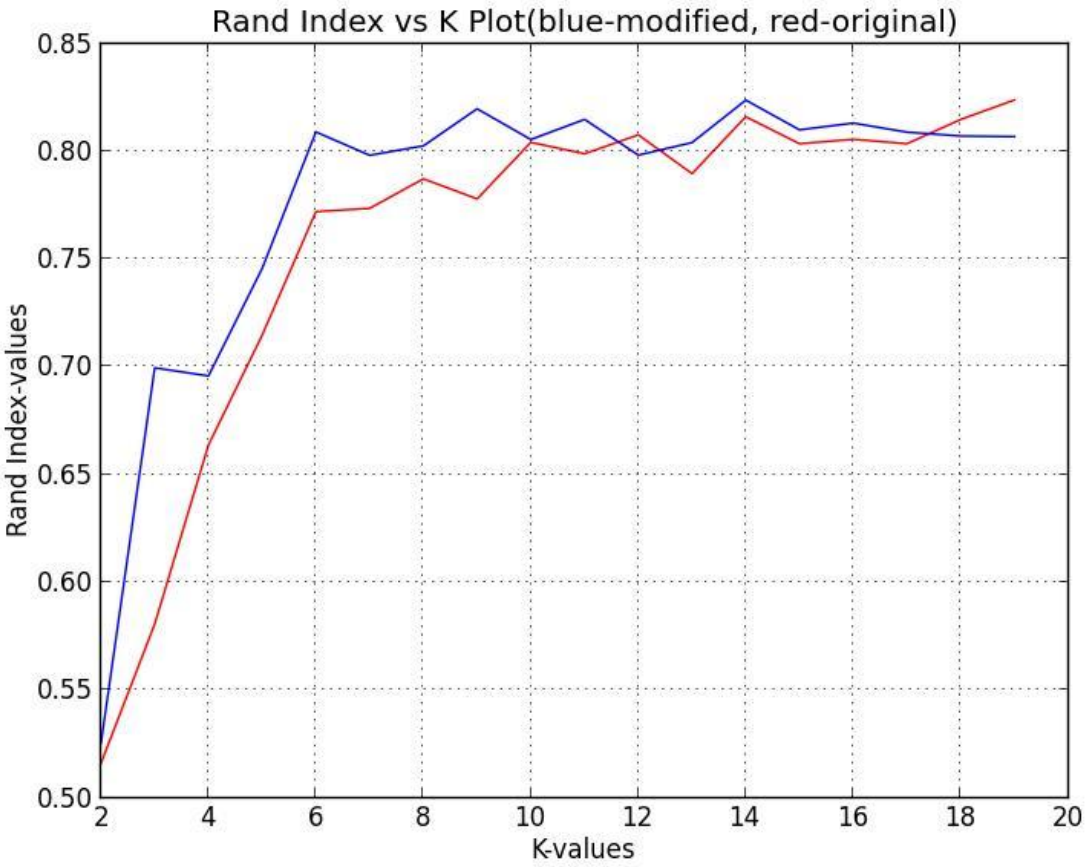
3. Proper selection of initial seeds/centroids: In part one the initial seeds are chosen randomly, so there might be a case in which clustering starts from the seeds of similar types, which obviously will hamper the result. However, in the improved version, I made sure that the initially chosen seeds are not of similar types.



In the improved version, the RSS values for different K-values have lowered down, which is an indication of improvement in clustering.



Even the purity has improved in the improved version as compared to the old version of clustering.



Rand Index values are also indicating the improvement in the newer version of clustering.

Modified Classification:

Improvement measures taken for Classification:

1. Removal of duplicate documents from the training set – Ideally, duplicate documents should not exist in the training set.

2. Removal of Stop Words – Stop words hardly adds to the relevance of any data. So, I eliminated all the stop words.

3. Feature selection though Mutual Information calculation – calculated mutual information of different terms with respect to different classes. Based on the obtained MI values, eliminated few terms from the corpus dictionary.

However, in spite of making these standard modifications, I couldn't see any enhancement in the result of Microaveraged F1.

Old Microaveraged F1 = 0.74888888888
New Microaveraged F1 = 0.726666666667

I think the reason might be the poor training data set.