PROJECT REPORT

AKSHAY KULKARNI
kulka182@umn.edu

PRE-PROCESSING

Following extra steps were taken during pre-processing

- Stop words were removed. Hard-coded list of stop words was included in code and tokens from the files were removed if they belonged in the list containing stop words.

- The words which occurred in less than **two** documents and in more than **90%** documents were removed. This is because words occurring very frequently or very rarely have any discrimination power.

- For each term in each document, its tf-idf was score was computed and to account for documents of different length, the length of each document vector was normalized.

- It takes around 4 minutes to process all the files across all folders generate all relevant files.

DATA STRUCTURES USED FOR PRE-PROCESSING

- **TRIE** data-structure was used to store different words across all documents. It helped in removal of duplicate words/tokens across different documents and kept only the unique ones.

- Dictionary(map) was used to store frequency of each token in the document and frequency of token across all documents for computing tf-idf score for each term.

DATA STRUCTURES USED FOR CLUSTERING

- Since Document-Term matrix is a high-dimensional matrix, it was stored in CSR format. High dimensional matrix can be stored in CSR format by using 3 arrays namely **rowind, colind and values.**

- **rowind** => rowind[i] stores the index of starting location of object i in colind.
- **colind =>** colind[i] stores the column index (or feature id) of the non-zero value.
- **values =>** values[i] stores frequency corresponding the feature stored at colind[i]

- **Centroids** were stored as dense vectors because centroids may have many non-zero values as its requires adding feature vectors of varying and different dimensions.

STATISTICS OF DATA AFTER PRE-PROCESSING

Feature Type	# Objects	#Dimensionality	#Non-Zeroes
Bag of words	6510	26939	605752
3-gram	6510	24515	2961108
5-gram	6510	170371	4724346
7-gram	6510	395860	4570502

STATISTICS FOR CLUSTERING

All tests were ran on CSE Lab machines

Feature Type	Clusters	Entropy	Purity	Time (in secs)
bag.csv	20	3.433860	0.249616	6.123669
bag.csv	40	3.297876	0.280799	12.339102
bag.csv	60	3.137308	0.306452	28.664572
char3.csv	20	3.577621	0.186329	24.968517
char3.csv	40	3.493217	0.229032	47.378383
char3.csv	60	3.250718	0.269739	81.006870
char5.csv	20	3.607473	0.212289	137.309805
char5.csv	40	3.187524	0.290169	251.974199
char5.csv	60	3.155056	0.296313	489.557144
char7.csv	20	3.547234	0.244086	155.251401
char7.csv	40	3.255730	0.282949	312.161432
char7.csv	60	3.185116	0.303072	437.415530

CLUTO paper suggests that using incremental k-means provide fast convergence and better results. Hence, I also implemented incremental k-means and observed convergence for incremental k-means was vey fast but there was only little improvement in the values of entropy and purity.

STATISTICS FOR CLUSTERING USING INCREMENTAL K-MEANS

Feature Type	Clusters	Entropy	Purity	Time (in secs)
bag.csv	20	3.197271	0.309985	4.388713
bag.csv	40	3.024575	0.350384	6.263792
bag.csv	60	2.893924	0.369124	10.037774
char3.csv	20	3.532771	0.221966	15.057917
char3.csv	40	3.205324	0.286329	24.142571
char3.csv	60	3.198336	0.285868	32.994624
char5.csv	20	3.293820	0.268971	63.641321
char5.csv	40	3.035446	0.338710	119.679837
char5.csv	60	2.957347	0.335791	164.163975
char7.csv	20	3.364369	0.267281	93.498168
char7.csv	40	3.098264	0.321045	152.125631
char7.csv	60	2.992384	0.342550	207.179542

In addition to this I have also plotted heat-map for clustering showing distribution of points from different class across various clusters. All code for this can be found here[https://goo.gl/LzgnT5]