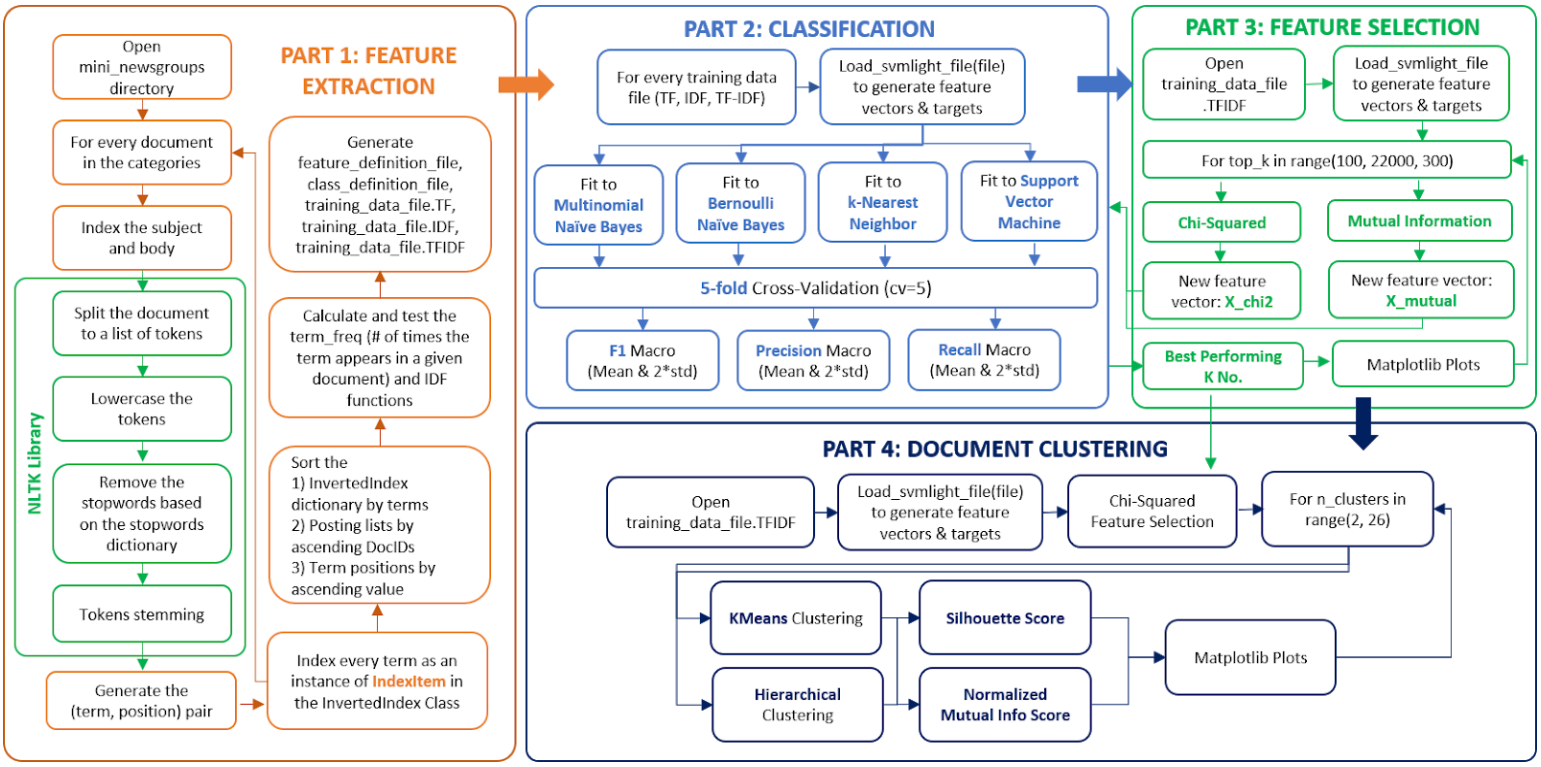
Spring 2019 - Information Retrieval (CS-7800-90)

GitHub Repository: https://github.com/Joeyipp/text-mining

Name: Hong Yung YIP (U00881791)

Design Flowchart & Implementation Pipeline



Instructions and Sample Commands

1. The program is implemented and tested on **Python 3.6.6**.
2. Included in the zipped source code directory is a **requirements.txt** that lists the external dependencies used. Do "**pip install -r requirements.txt**" prior to running the scripts.

Experimental Results and Discussions

PART 1: EXTRACTING FEATURES

RUN: python3 feature-extract.py mini_newsgroups feature_definition_file class_definition_file training_data_file

```
-> Indexing /home/joey/Desktop/Project 2/mini newsgroups/soc.religion.christian
100%|

-> Indexing /home/joey/Desktop/Project 2/mini newsgroups/rec.sport.baseball
100%|

-> Indexing /home/joey/Desktop/Project 2/mini newsgroups/rec.autos
100%|

-> Indexing /home/joey/Desktop/Project 2/mini newsgroups/talk.politics.misc
100%|

-> Indexing /home/joey/Desktop/Project 2/mini newsgroups/comp.sys.mac.hardware
100%|

Total documents indexed: 2000

-> Generating training data files
100%|

-> Generating feature definition file
Done!

-> Generating class definition file
Done!

-> All done!
```

Figure 1A: Sample Script Output

```
-> Testing Class Definition
{'comp.graphics': 1, 'comp.os.ms-windows.misc': 1, 'comp.sys.ibm.pc.hardware': 1, 'comp.sys.mac.hardware': 1, 'comp.windows.x': 1, 'rec.autos': 2, 'rec.motorcycles': 2,
 'rec.sport.baseball': 2, 'rec.sport.hockey': 2, 'sci.crypt': 3, 'sci.electronics': 3, 'sci.med': 3, 'sci.space': 3, 'misc.forsale': 4, 'talk.politics.misc': 5, 'talk.p
olitics.guns': 5, 'talk.politics.mideast': 5, 'talk.religion.misc': 6, 'alt.atheism': 6, 'soc.religion.christian': 6}

-> Indexing C:\Users\User\Desktop\text-mining\mini_newsgroups\alt.atheism\51121
---> Class Label: 6
---> Document ID: 51121
---> Document Body:
 Re  [soc.motss, et al.] "Princeton axes matching funds for Boy Scouts"  In article <N4HY.93Apr5120934@harder.ccr-p.ida.org>, n4hy@harder.ccr-p.ida.org (Bob McGwier) wr
ites:  |> [1] HOWEVER, I hate economic terrorism and political correctness |> worse than I hate this policy.   |> [2] A more effective approach is to stop donating |> t
o ANY organizating that directly or indirectly supports gay rights issues |> until they end the boycott on funding of scouts.  Can somebody reconcile the apparent contr
adiction between [1] and [2]?  -- Rob Strom, strom@watson.ibm.com, (914) 784-7641 IBM Research, 30 Saw Mill River Road, P.O. Box 704, Yorktown Heights, NY  10598

-> Parsed Document Terms (Tokenized -> LowerCase -> Stopwords Removed -> Stemmed)
---> ['soc', 'motss', 'et', 'al', 'princeton', 'axe', 'match', 'fund', 'boy', 'scout', 'articl', 'n4hi', '93apr5120934', 'harder', 'ccr', 'p', 'ida', 'org', 'n4hi', 'ha
rder', 'ccr', 'p', 'ida', 'org', 'bob', 'mcgwier', 'write', '1', 'howev', 'hate', 'econom', 'terror', 'polit', 'correct', 'wors', 'hate', 'polici', '2', 'effect', 'appr
oach', 'stop', 'donat', 'organiz', 'direct', 'indirect', 'support', 'gay', 'right', 'issu', 'end', 'boycott', 'fund', 'scout', 'somebodi', 'reconcil', 'appar', 'contrad
ict', '1', '2', 'rob', 'strom', 'strom', 'watson', 'ibm', 'com', '914', '784', '7641', 'ibm', 'research', '30', 'saw', 'mill', 'river', 'road', 'p', 'box', '704', 'york
town', 'height', 'ny', '10598']

Total documents indexed: 1

-> Term: Feature ID: TF
soc
FID: 1, TF: 1
```

Figure 1B: Test Cases for class_definition_file, indexing, document parsing, and term frequency

Results:

Table 1: Dataset Statistics

Number of newsgroups categories	20
Documents per newsgroups category	100
Total Number of Training Data/ Instances	2000
Feature Size	32715
Ratio of features to training instances	16:1

Discussions:

An inverted index (Figure 1A) was built to extract **three** distinct features: Term Frequency (**TF**), Inverse Document Frequency (**IDF**), and Term Frequency-Inverse Document Frequency (**TF-IDF**) from the mini newsgroups with 20 different news categories from 6 classes. Each row in the **class_definition_file** is a mapping of (newsgroup, class_label). Whereas **feature_definition_file** contains the list of terms used as features. Each line contains a pair (feature id, term) sorted by index. There were three training data files generated from running feature-extract.py: training_data_file.TF, training_data_file.IDF, and training_data_file.TFIDF in the libsvm format (the sparse representation of the document-term matrix). The documents (training data) are randomly shuffled before writing to file to optimize the classifiers' training. Each row represents one document and takes the form < class label> <feature-id>:<feature-value> <feature-id>:<feature-value>. Table 1 showed a relatively high 16:1 ratio of the number of features to training data.

PART 2: CLASSIFICATION

RUN: python3 classification.py

Results:

```
(base) joey@knoesis-380:~/Desktop/Project 2$ python3 classification.py

-> Classifiers Evaluation using training data file.TF

                    F1 Macro              Precision Macro         Recall Macro
Multinomial NB      0.77 (+/- 0.04)       0.84 (+/- 0.09)         0.77 (+/- 0.03)
Bernoulli NB        0.52 (+/- 0.02)       0.62 (+/- 0.02)         0.51 (+/- 0.03)
kN-Neighbor         0.38 (+/- 0.12)       0.62 (+/- 0.09)         0.37 (+/- 0.09)
SV-Machine          0.08 (+/- 0.01)       0.33 (+/- 0.25)         0.17 (+/- 0.01)

-> Classifiers Evaluation using training data file.IDF

                    F1 Macro              Precision Macro         Recall Macro
Multinomial NB      0.83 (+/- 0.05)       0.84 (+/- 0.05)         0.83 (+/- 0.05)
Bernoulli NB        0.52 (+/- 0.02)       0.62 (+/- 0.02)         0.51 (+/- 0.03)
kN-Neighbor         0.21 (+/- 0.13)       0.44 (+/- 0.24)         0.27 (+/- 0.10)
SV-Machine          0.07 (+/- 0.00)       0.04 (+/- 0.00)         0.17 (+/- 0.00)

-> Classifiers Evaluation using training data file.TFIDF

                    F1 Macro              Precision Macro         Recall Macro
Multinomial NB      0.83 (+/- 0.03)       0.83 (+/- 0.03)         0.85 (+/- 0.03)
Bernoulli NB        0.52 (+/- 0.02)       0.62 (+/- 0.02)         0.51 (+/- 0.03)
kN-Neighbor         0.30 (+/- 0.12)       0.62 (+/- 0.13)         0.31 (+/- 0.08)
SV-Machine          0.10 (+/- 0.02)       0.51 (+/- 0.30)         0.18 (+/- 0.01)

-> All done!
```

Figure 2: Mean Classification F1, Precision, and Recall Macro on TF, IDF, and TF-IDF Features

Discussions:

Four classification algorithms: **Multinomial** Naive Bayes (NB), **Bernoulli** NB, k-Nearest Neighbor (**kNN**), and Support Vector Machine (**SVM**) were experimented on **three** types of feature: TF, IDF, and TF-IDF using the **default parameters**. Each classifier was then evaluated using **5-fold cross validation** on the whole dataset to compute a reliable estimation of the classifier performance. The mean and 2*std of 5-fold F1, Precision, and Recall Macro were shown on Figure 2 respectively.

Based on Figure 2, all three features showed relatively similar results with **TF-IDF** feature as the most optimum feature. **Multinomial NB performed the best** (in terms of F1, precision, and recall macro-averaging for different classes) in the overall classification task with an F1 Macro of **0.83** or **83%** with low standard deviation, followed by Bernoulli NB (0.52), kNN (0.30), and lastly SVM as the least performing classifier with only 10% F1 Macro. **With high number of features, an equally high number of training data are needed** for the classifier/ regressor to correctly "learn" the importance and weight of each feature. High ratio of features to training data (16:1) in this experiment, with 32715 features and only 2000 total training instances (100 documents per 20 news categories) increases the possibility of model **overfitting** (further illustrated in the Part 3) in which the model failed to generalize the data trend.

Nonetheless, Multinomial NB is designed for classification with discrete features (e.g., word counts for text classification), hence it was the highest performing model. Whereas for Bernoulli NB, it is also suitable for discrete data. However, the difference is that while Multinomial NB works well with occurrence counts such as TF, IDF, or TF-IDF, Bernoulli NB is designed for binary/boolean (0 or 1) features, which is **not the type of feature this experiment was using**. Hence, this explains Bernoulli NB as the second best performing model. A very high dimension feature space in kNN (default 5K) with Euclidean distance leads to the problem that the training samples will be too sparse across the space and render all classes virtually indistinguishable, which explains the poor performance. On the contrary, SVM does not depend on the number of training data since only the support vectors are required to construct the separating hyperplane, and therefore theoretically, SVM should achieve high performance. Some of the plausible reasons are the default kernel (Radial basis function) and parameters used to transform the data into linearly-separable problem is sub-optimal and number of features space is too high and sparse. This experiment concluded that using all 32715 features yielded poor performances. In the next section, a feature selection/ reduction was applied to search for the best performing number of K-features and how the K-number affected the overall classifiers performance.

PART 3: FEATURE SELECTION

RUN: python3 feature_selection.py

Results:

Table 2: Best Performing Number of K-Features for each Classifiers

Classifiers	Chi-Squared	F1 Macro (%)	Mutual Info.	F1 Macro (%)
Multinomial Naive Bayes	8500	**90.76**	11500	**89.24**
Bernoulli Naive Bayes	700	**76.43**	2200	**75.93**
k-Nearest Neighbours	400	**64.17**	100	**60.92**
Support Vector Machine	400	**65.30**	400	**69.79**

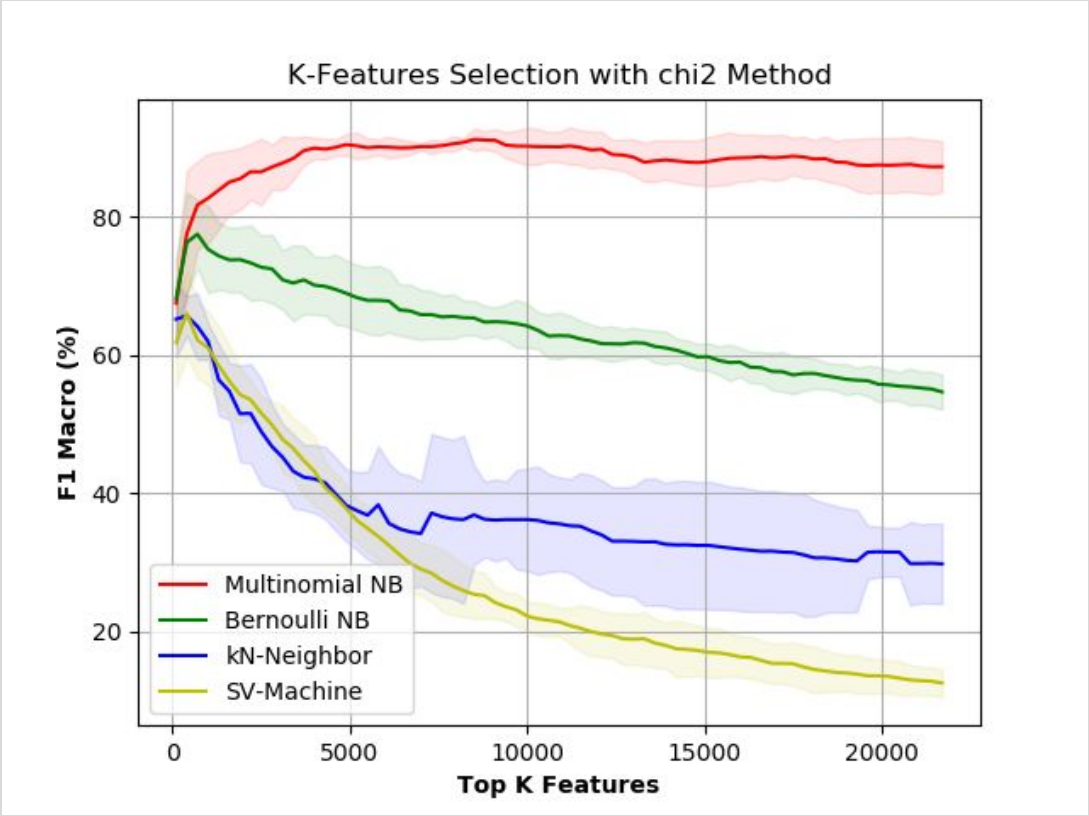


Figure 3: Features Selection with K ranging from 100 to 22000 using Chi-Squared Method

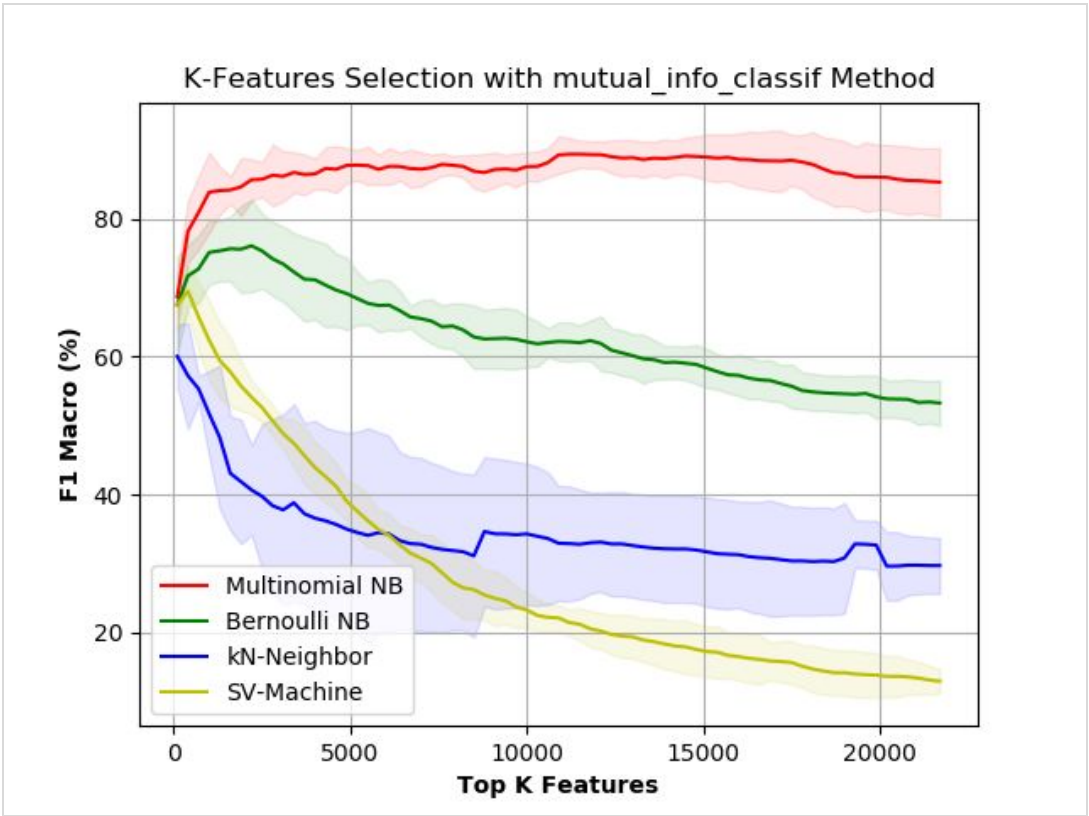


Figure 4: Features Selection with K ranging from 100 to 22000 using Mutual Information Method

Discussions:
Since Part 2 of the experiment established that there were relatively marginal differences between the three different features (TF, IDF, and TF-IDF), for the subsequent parts (Part 3 and Part 4), all experiments were conducted using **only TF-IDF feature**. Two feature selection methods were adopted: the **Chi-Squared** and the **Mutual Information** method. A number of K-features ranging from 100 to 22000 with a step-size of 300 were tested. Table 2 tabulated the best performing number of K-features and the performance of each classifiers respectively.

All four classifiers achieved a substantial improvements from using an optimum number of features, with an average F1 Macro of 60% and above. It was observed that the upper-bound number of K-features using both feature selection methods for the best performing classifier, Multinomial Naive Bayes were approximately 10,000 features respectively and any increment beyond 10,000 features yielded little to no improvements. On the contrary, the performance of the Bernoulli Naive Bayes, kNN, and SVM declined as the number of feature increases. Figure 3 and figure 4 illustrated that **high features to training data ratio degraded the classifiers' performance.**

This observation validated the initial hypothesis that the high ratio of features to training data led to (i) model overfitting, and (ii) sparse features space renders all classes virtually indistinguishable. A high number of features space should have an equally high number of training data for the classifiers to "learn" the importance and pattern of each feature.

PART 4: DOCUMENT CLUSTERING

RUN: python3 clustering.py

Results:

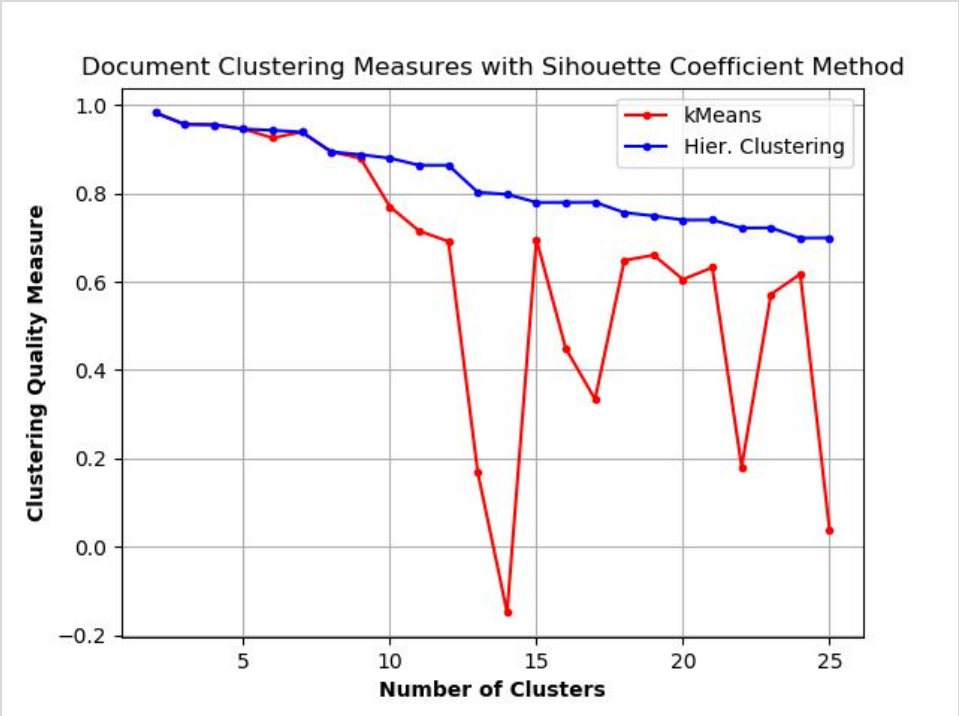


Figure 5: kMeans and Hierarchical Clustering Quality Measures with Silhouette Coefficient Method

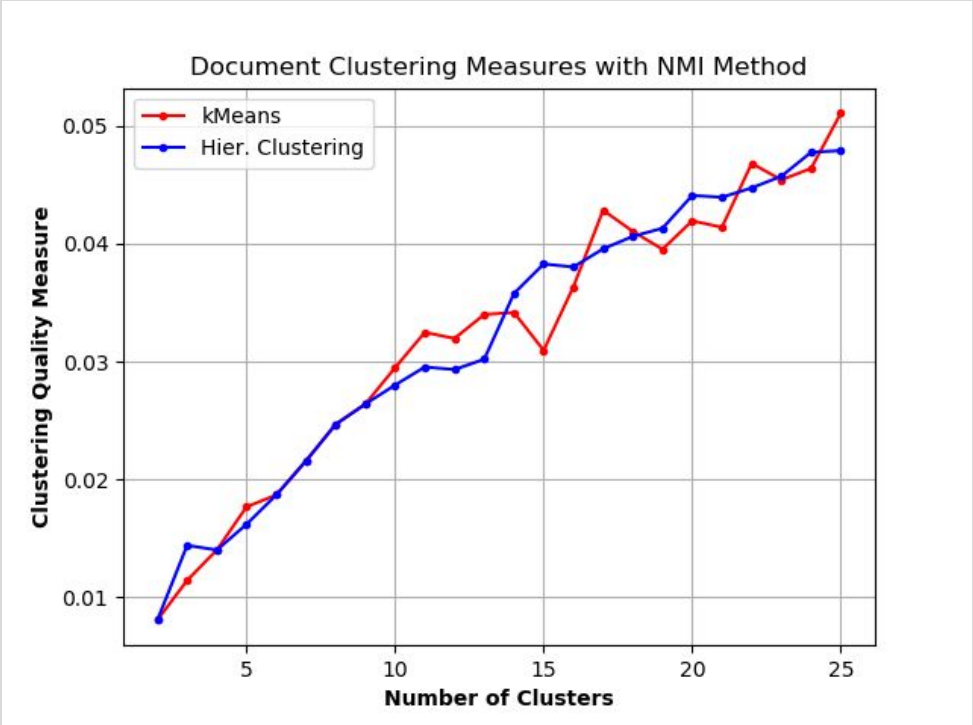


Figure 6: kMeans and Hierarchical Clustering Quality Measures with Normalized Mutual Information (NMI) Method

Discussions:

Figure 5 and 6 showed the clustering quality measure for both kMeans and Hierarchical clustering using **Silhouette Coefficient (SC)** and **Normalized Mutual Information (NMI)** methods on 10,000 TF-IDF features using Chi-Squared selection method since any increment beyond 10,000 features yielded poor performance (Figure 3, 4).

The SC is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. Overlapping clusters have a value of near zero, whereas negative values generally indicate that a sample has been mis-clustered (wrong cluster). The NMI is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation). The metric is independent of the absolute values of the labels. A score of 1.0 indicates perfect labelings for both homogeneous and complete clusters. Whereas, if classes members are completely split across different clusters, the NMI is near 0.

Based on Figure 5 and 6, it can be observed that, using Hierarchical clustering yielded a more consistent clustering quality compared to kMeans clustering. Figure 5 showed the Hierarchical clustering has a steady decline in clustering quality as the number of clusters increased, which is concordance to the experimental training data with only 6 class labels and 20 categories. Similar news documents formed a similar news category, and similar news categories formed a class, hierarchically. On the contrary, kMeans clustering was unstable as the number of clusters exceeded 10. One possible reason is the similarity between the news documents forming overlapping clusters.

Figure 6 showed that the value of NMI increases as the number of clusters increases. Such observation can be simply explained as the higher the number of clusters, the higher the chance of clusters overlapping and hence sharing mutual information (higher NMI value).