### INFORMATION RETRIEVAL PROJECT

### TITLE: Text Mining

### Project Members:

### Sai Sri Malempati (U00973516)

### Srivalli Kompella (U00965253)

### Part 1: Extracting features

**Design:**

1. Read all the files from mini\_newsgroups directory.
2. Read subject and last xx lines mentioned in Lines: xx of the files as title and body of the documents
3. Create an Inverted Index for documents.
4. Create a class\_definition\_file with newsgroup name and class\_group\_id.
5. Create a feature\_definition\_file with index terms and feature ids as (feature id, term) for each term in index.
6. Create a training\_data\_files in the libsvm format using term frequency, inverse document frequency, tf-idf. The term frequency , IDF ,tf-idf are calculated after index file generated .

**Implementation:**

we have implemented Extracting feature using classes feature\_extract, Index in feature\_extract.py , read\_news in news.py. Read all the files from mini\_newsgroups directory and load the documents using doc.py by reading subject and last xx lines mentioned in Lines of the file. The read\_news class have a class variable, docs which is list of Document. Iterate these document list and form the index object using Index class method add(). We have removed stopwords by using **stopwords from nltk.corpus** . For Stemming we have used **PorterStemmer from NLTK** . Then for each of these stemmed words we must calculate the position of occurrence in the document. The index object has terms and list of document ID’s it occurred. Create a hard-coded variable as class\_map\_dic with key as Class ID and value as list of newsgroups belong to that class ID. Write this variable to the file class\_definition\_file. This entire logic is implemented in load\_class\_definition\_file() method. Create an Feature ID for each term in index object and write this (feature ID, term) in feature\_definition\_file. This entire logic is implemented in load\_feature\_definition\_file() method. Load the training\_data\_file.TF and training\_data\_files.IDF, training\_data\_files.TFIDF by getting the document class ID from class\_definition\_file and for the terms in those documents get their feature ID, value (this could be TF,IDF,TF-IDF based on file creation). Form the string for each document and write them to the corresponding file. This logic is implemented in load\_training\_data\_file\_TF() method.

**Formulas for TF,IDF,TF-IDF are as follows:**

### TermFrequency(t)= The Frequency (Number of times) of a term ‘t’ Occurs in the Document

### Inverse Document Frequency (IDF) = Log(Total Number of Documents/Number of Documents Contains the Term ‘t’)

### TF-IDF = TermFrequency \* Inverse Document Frequency

**Run the feature-extract.py** ::: python feature-extract.py mini\_newsgroups feature\_definition\_file class\_definition\_file training\_data\_file.

A screenshot of a social media post

Description automatically generated

Output of class\_definition\_file:

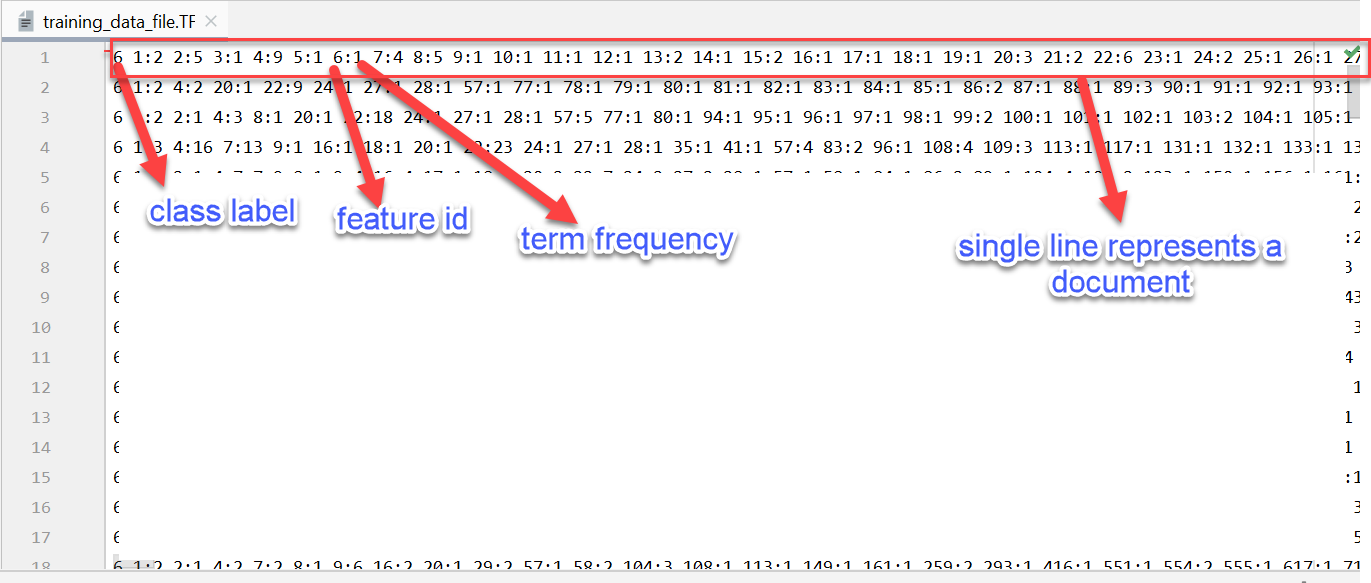
A close up of a map

Description automatically generatedOutput of feature\_definition\_file:

A screenshot of a social media post

Description automatically generated

Output of training\_data\_file.TF:



Output of training\_data\_file.IDF:

A screenshot of a social media post

Description automatically generated

Output of training\_data\_file.TFIDF:

A close up of a map

Description automatically generated

**Test Cases:**

* To check logic works to load all files of input directory
* Testing Index creation , removal of stopwords ,stemming on a document
* Testing feature\_defintion\_file, class\_definition\_file created.
* Testing load of TF,IDF,TFIDF training\_data\_files into sklearn.

Python run : python test.py

A screenshot of a social media post

Description automatically generated

**Part** **2:** **Classification**

**Design:**

### we apply different classification algorithms on the training\_data\_files created in part 1. We use algorithms like Multinominal Naive Bayes, Bernoulli Naive Bayes, k Nearest Neighbor, and Support Vector Machines. split the data for test and train using cross\_val\_score and use metric as f1\_macro, which is the macro-averaging of the F1 scores. Calculate the mean and 2\*std of 5-fold f1\_macro, precision\_macro, and recall\_macro for all the classification algorithms. A good classifier has F1 score greater than 0.5. precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples. The F-measure can be interpreted as a weighted harmonic mean of the precision and recall.

### Precision = True Positives / (True Positives + False Positives)

### Recall = True Positives / ( True Positives + False Negatives)

### F1 = 2 \* (precision \* recall) / (precision + recall)

**Implementation:**

We have loaded the training\_data\_files using load\_svmlight\_file() method which provide features and target. For multinomial algorithm, we have used training\_data\_file.TF and loaded feature vectors and targets for this using load\_svmlight\_file. Using cross\_val\_score function and calculated scores means and standard deviations for f1\_macro, precision\_macro, and recall\_macrofor all the classification algorithms.

**Observations and Experimental Results:**

From the observation we found that, Multinomial Naïve Bayes classification have 0.65 approximately F1 score which is a higher value greater than 0.5. Naive Bayes , K-nearest neighbors classification have around 0.34 f1\_score which shows not a good classifier for the dataset. C-support vector classifier have 0.43 f1\_score which is shows not a good classifier for the dataset. Among all Multinomial Naïve Bayes classification is reasonable best classifier for the dataset.

Python run: python classification.py

A screenshot of a social media post

Description automatically generated

### Part 3: Feature Selection

**Design:**

The Chi-square and the Mutual information method are used to find whether the features are relevant or not. We can remove non-relevant features that are not valuable to produce the outcomes of the model. These methods are called as Feature Selection methods as they only select the feature which are relevant. SelectKBest is used to remove all the features except ‘k’ mentioned features which are the high scored among all. We apply the chi-square and mutual information methods for theses K features and calculate the F1 macro scores which is considered as accuracy.

**Implementation:**

Consider k-values ranging from 500 to 2000. Here we have taken 500, 700, 900, 1100, 1300, 1500, 1700, 1900 as k-values. For each k-value, load the features and target of all the four algorithms [Multinominal Naive Bayes, Bernoulli Naive Bayes, k Nearest Neighbor, and Support Vector Machines] and select k-best features using SelectKBest function from sklearn.feature\_selection package. Then for each algorithm calculate the f1\_macro scores for both chi-square and mutual information methods. Then Plot the graphs for chi-square and mutual information methods for all the classifiers as x-axis: k values and y-axis: f1\_macro measures.

Python run: python feature\_selection.py

A screenshot of text

Description automatically generated

**Observations and Experimental Results:**

The Chi Square graph shows CHI-square technique of feature selection for multinomial, Bernoulli ,KNN,SVM classifiers of f1\_macro measure at different k values [k-value is the ‘k’ best features selected]. From graph it shows that applying CHI Square method have a reasonably good accuracy on Multinomial, Bernoulli classifiers as they have the accuracies [f1\_macro] measure greater than 0.5 at all values of K. The KNN,SVM shows low accuracy <0.5 for chi-square feature selection method at all values of K.A close up of a map

Description automatically generated

Mutual Information graph shows that Mutual Information technique of feature selection for multinomial, Bernoulli ,KNN,SVM classifiers of f1\_macro measure at different k values [k-value is the ‘k’ best features selected]. From graph it shows that applying Mutual information method have a reasonably good accuracy on Multinomial, Bernoulli classifiers as they have the accuracies [f1\_macro] measure greater than 0.5 at all values of K. The KNN,SVM shows low accuracy <0.5 for chi-square feature selection method at all values of K. The chi-square shows a decreasing trend in Multinominal classifier whereas mutual information shows an increasing trend for the same classifier as k values increases.

A screenshot of text

Description automatically generated**Part 4: Document Clustering**

**Design:**

For document clustering, instead of taking all the features we can consider smaller K best features selection and perform the two clustering algorithms: kMeans and hierachical clustering. KMeans clustering algorithm is performed by grouping the similar documents or features using centroids calculations. Hierarchical clustering is performed by grouping the documents or features in a hierarchical manner top to bottom or bottom to top manner. The hierarchical clustering algorithm uses the agglomerative style (bottom-up) and one of the three linkage methods (ward, complete, average) for computing cluster distances. The clustering algorithms quality is measured by Sihouette Coefficient (SC) and Normalized Mutual Information (NMI).

**Sihouette Coefficient (SC)**= (b-a)/max(a, b)

where a= Average of distance between a point and all other points within a cluster , b= Avergae of distance between a point and all other points in the nearest cluster.

**Normalized Mutual Information (NMI)** = (2 \* I (Y;C))/(H(Y)+ H(C)]

where, Y = labels of class ; C = labels of cluster ; H(.) = Entropy

(Y;C) = Mutual Information b/w Y and C

**Implementation:**

Load the training\_data\_file.TFIDF using load\_svmlight\_file() method to features and targets. Select top 100 best features using SelectKBest method and convert to an array. Take a range of clusters from 2 to 25. For each cluster, run the k-means algorithm and hierarchical clustering (agglomerative) and perform the quality test by calculating Sihouette Coefficient (SC) and Normalized Mutual Information (NMI) using metrics.silhouette\_score() and metrics.normalized\_mutual\_info\_score() .The plot the graphs for all clusters on X-axis and measure of silhouette and normalized\_mutual\_info

Python Run: python clustering.py

A screenshot of a cell phone

Description automatically generated

# A screenshot of a computer Description automatically generated

**Observations and Experimental Results: A close up of a map

Description automatically generated**

**A close up of a map

Description automatically generated**