README

This ReadMe document will describe in detail how to execute the program and also give a brief description of all the files and functions in the code.

Requirements: Python 3+, NLTK library (For text manipulations)

Steps to install NLTK:

* *$* sudo pip install –U nltk
* *$* sudo pip install –U numpy
* Run python on the command line
* *$* python
* *>>* import nltk
* *>>* nltk.download()
* Install from the client that opens.

Once the dependencies are downloaded the program can be executed using the command:

*$* python ml\_naive\_bayes.py

It takes around 15-18 minutes for all the files to be processed and final accuracy to be displayed.

The zip contains a data directory which contains all the newsgroup data used for training and testing the algorithm. A ReadMe file which describes how to use the program and a python executable file called ml\_naive\_bayes.py

Description of ml\_naive\_bayes.py

It contains the following functions:

* data\_splitter

This function splits the files text part from the headers. Only the text part is used for training and testing purpose.

* tokenizeString

It uses nltk to identify different words in the text and divide the text into tokens. A token for each word.

* createVector

It is used to count the frequency of each word in the tokenized string store the values in a vector

* getArgMax

This function calculates the probability for each class on the testing document and returns the class with maximum probability as the predicted class

* Main function steps:
  + Save all file names in the data directory to be used as training and testing data
  + Using half of the files in each class as training data, use above described methods to tokenize the string and find the frequency of each word in the documents.
  + Get total number of words in all the documents for a class
  + Get total number of distinct words in all the training documents
  + Training phase: calculate probability of each word given a class and store these values
  + Get total docs and use it to calculate prior probabilities for each class
  + Use other half of the documents for testing
  + Testing: using prior probabilities and probabilities of each word given a class, choose a class with maximum probability. This will be the predicted class to which the document should belong
  + Calculate accuracy based on the actual and predicted classes