Data Mining

Aakash Rohilla | 1001167363

Home Work Assignment 5

Import Guidelines for running BayesianClassifier.py:

1. "articles" folder consists of two subfolders:
    a. Training
       Contains first 150 files of "arxiv", "jdm" & "plos" folder. These files will be used as training data.
    b. Testing
       Contains last 150 files of "arxiv", "jdm" & "plos" folder. These files will be used as testing data.
2. "BayseianClassifier.py" is the python file which is responsible for using training data in order to compute the probabilities that will be used later on in order to make prediction about the class of an article.

Functions in BayesianClassifier.py:

This section explains all the important functions that are used in order to make predictions about articles and compute the prediction accuracy.

1. **calculateWordFrequencyinClass(folderPath)**:
   This function is responsible for reading a training folder "arxiv", "jdm" & "plos" and counting the frequency of all the words for all the articles in that folder. This function makes sure that the words in "stoplist.txt" are not moved to the vocabulary for that class, the vocabulary consists of words and their number of occurences in that class.

2. **mergeDictionaries():**
   This function take vocabulary of "arxiv", "jdm" & "plos" and combines them to make a single vocabulary. This includes adding frequencies of words that are occurring in more than one class, also including words that are only occurring in a single class.

3. **countTotalWordsInAClass(className):**
   This function counts the total number of words for a given class. className can be anything of the follows: "arxiv" or "jdm" or "plos".

4. **generateProbabilityOfWords(totalWordCount):**
   This function takes total word count, which is of all the words in the merged vocabulary and generates probability of a given word. It is computed using the formula:

   Probability(Word) = (Number of occurrences of that word)/(Total number of words in vocabulary)

5. **generateProbabilityOfWordGivenAClass(className):**
   This function generates the probability of a given word in a class. The input to this function is the name of the Class i.e "arxiv" or "jdm" or "plos".
   . This is done using the following formula.

   Probability(Word) = (Number of occurrences of that word in that class)/(Total number of words in vocabulary in that class)

6. **generateProbOfClassGivenAWord(className):**
   This function computes the probability of class given a word. The input to this function is the name of the Class i.e "arxiv" or "jdm" or "plos". This probability is calculated using the following formula:

   *Probability(Class|Word) = (Probability(Word|Class)*Probability(Class))/Probability(Words)*

   Where, Probability(Class) = 1/3, because we have 3 classes and any one of them could be chosen. Rest of the parameters have already been calculated using the above functions and hence their results can be used in order to get the result for this equation.

7. **probabilityOfClassForAnArticle(folderPath):**
   This function takes as input the folder path of the testing data. This folder path can be any of the 3 i.e "arxiv" or "jdm" or "plos". Once we have the article from a folder, for each and every word we compute the probability of class using the formula.

   *Probability(Class) = Probability(Class|Word)*Probability(Word)*

   We do this for each and every word in the article and for all the 3 classes in question. We sum the probabilities for each class. Now, in order to make the prediction we check which Class has the highest probability. The class that has the highest probability is the the predicted class to which the article belongs. This is done for all the articles in the class folders, in the testing data. The final output of these files is written to <ClassName>OutputFile.txt. A snapshot of the these files is given below.

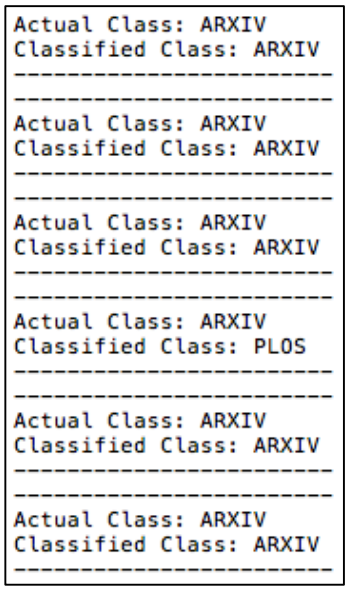

Fig (a) arxivOutputFile.txt

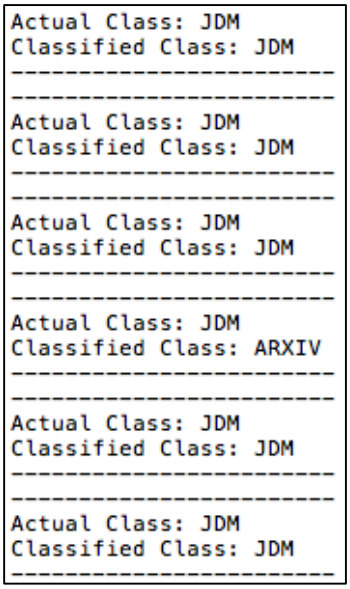

Fig (b) jdmOutputFile.txt

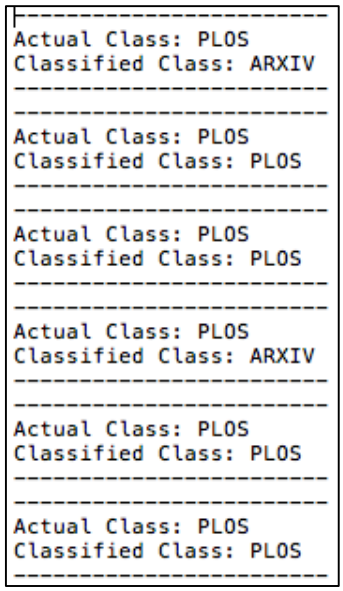

Fig (c) plosOutputFile.txt

8. **writeToFile(dataForFile,fileName):**
   Writes string data to a particular file and saves it as ".txt" file. It takes two arguments, first is *dataForFile* which is string content that needs to be written to the file. Second is *fileName* which is the name of the file to be given to the file.