CS 453, Fundamentals of Information Retrieval, Winter 2013

Project Assignment 3

Text Document Classification Using a Multinomial Naïve Bayes Model

due Tuesday, April 2

1 Project Description

The purpose of this project assignment is to (i) implement and apply the *Multinomial Naïve Bayes* (*MNB*) model for text document classification and (ii) perform *feature selection* on the vocabulary of a given document collection to reduce the training and test time of the MNB classifier without significantly affecting its accuracy.

You can implement this project in any programming language of your choice; however, you must follow the implementation specifications given in Section 2.

2 Implementing the Multinomial Naïve Bayes Classifier

In this project, you are required to implement the MNB classifier, using the classes and methods specified in this section.

2.1 The MNB-$classification$ class

This class is used for (i) training an MNB and (ii) classifying unlabeled documents using the trained MNB.

Given a document collection DC, which is a set of plain-text documents, all the stopwords in DC are first removed. (The list of stopwords to be used is available at http://students.cs.byu.edu/˜cs453ta/projs/stopwords.) Thereafter, DC should be partitioned into two subsets: the $DC\_training$ set, which contains 80% of the documents randomly selected from DC, and the DC_test set, which contains the remaining 20% of the documents in DC. If you are using the entire vocabulary, i.e., when no *feature selection* is applied, then a data structure *training_set* (*test_set*, respectively), which includes the multinomial representation of each document in the $DC\_training$ (DC_test, respectively) set should be created. (See a sample multinomial representation on Slide #15 in the Lecture Notes of Chapter 9). On the other hand, if *feature selection* is applied, then the documents in the $DC\_training$ set should first be used to determine the *selectedFeatures* (as defined in the *featureSelection* method in Section 2.1.1). Hereafter, *training_set* (*test_set*, respectively), which consists of the multinomial representation of each document in the $DC\_training$ (DC_test, respectively) set using only the terms in *selectedFeatures*, should be created.

2.1.1 The $featureSelection$ method

This method determines the words which should be chosen to represent documents in *training_set* and *test_set* based on the *information gain* (IG) of each vocabulary[1] term w, which is computed as follows:

$$IG(w) = -\sum_{c \in C} P(c)log_2P(c) + P(w)\sum_{c \in C} P(c|w)log_2P(c|w) + P(\bar{w})\sum_{c \in C} P(c|\bar{w})log_2P(c|\bar{w}) \quad (1)$$

[1]A *vocabulary* consists of all the distinct words in the training portion of a document collection used for training an MNB model, i.e., the *DC_training* set in this case.

where C is the set of distinct natural classes in DC, $P(c)$ is the probability of the natural class c, $P(w)$ ($P(\bar{w})$, respectively) is the probability of occurrence (absence, respectively) of w, and $P(c|w)$ ($P(c|\bar{w})$, respectively) is the probability of c given that w is present (absent, respectively) in c.

$$P(c) = \frac{\text{\# of documents in } DC\_training \text{ labeled as } c}{\text{Total number of documents in } DC\_training}$$

$$P(w) = \frac{\text{\# of documents in } DC\_training \text{ in which } w \text{ occurs}}{\text{Total number of documents in } DC\_training}$$

$$P(\bar{w}) = \frac{\text{\# of documents in } DC\_training \text{ in which } w \text{ is absent}}{\text{Total number of documents in } DC\_training}$$

$$P(c \mid w) = \frac{\text{\# of documents in } DC\_training \text{ in which } w \text{ occurs that are labeled as } c}{\text{Total number of documents in } DC\_training \text{ in which } w \text{ occurs}}$$

$$P(c \mid \bar{w}) = \frac{\text{\# of documents in } DC\_training \text{ in which } w \text{ is absent that are labeled as } c}{\text{Total number of documents in } DC\_training \text{ in which } w \text{ is absent}}$$

The input of $featureSelection$ includes the *DC_training* set and a user-defined value M (≥ 1) which denotes the *number of words* in the vocabulary that should be selected. (If M = |Size of the Vocabulary|, then no feature selection is applied. Otherwise, M is one of the values specified in Section 3.) The output of $featureSelection$ is a data structure with M words, denoted $selectedFeatures$, which are the words in the vocabulary with IG values higher than the other words in the *DC_training* set and are used to represent each document in *training_set* and *test_set*.

2.1.2 The *label* method

This method assigns the most probable class for a particular document in *test_set*. In performing the classification task, you are required to use the *getWordProbability* and *getClass-Probability* methods, which are defined in Sections 2.2.3 and 2.2.4, respectively.

The input of *label* is a document D in *test_set*, and the output is the class c that should be assigned to D.

2.2 The *MNB-probability* class

This class is used for training an MNB.

2.2.1 The *computeWordProbability* method

This method computes the *probability* of each distinct *word* in each natural *class* in C using *training_set*. In computing the word probabilities, you are required to use the *Laplacian Smoothed Estimate*. (See Slide #16 in the Lecture Notes.)

The input of *computeWordProbability* is *training_set*, and the output is a data structure called "WordProbabilities," which includes for each word in the vocabulary its probability for each class in C.

2.2.2 The *computeClassProbability* method

This method computes the *probability* of each natural *class* in C using *training_set*.

The input of *computeClassProbability* is *training_set*, and the output is a data structure called "ClassProbabilities," which includes the *probability* of each class in C.

2.2.3 The *getWordProbability* method

This method *retrieves* the *probability* value of a *word* in a particular class, which includes the probability value of each word not seen during the training phase of MNB.

The input of *getWordProbability* includes a word w and a class c, and the output is the *probability* of w in c, which is stored in "WordProbabilities."

2.2.4 The *getClassProbability* method

This method *retrieves* the *probability* value of a natural *class.*

The input of *getClassProbability* is a class c, and the output is the *probability* of c, which is stored in "ClassProbabilities."

2.3 The *MNB-evaluation* class

This class computes the *accuracy* of the trained MNB in text document classification.

2.3.1 The *accuracyMeasure* method

This method computes the *accuracy* of the trained MNB. **Accuracy** is defined as the *proportion* of documents in *test_set* for which their classification labels determined using the trained MNB are the same as their pre-defined, i.e., original, labels over the total number of documents in *test_set*.

The input of *accuracyMeasure* is the set of documents in *test_set* and their labels determined by using the method *label* in *MNB-classification*, whereas the output is the *classification accuracy* of the documents in *test_set*.

3 Classification

Having implemented the MNB classifier following the specifications in Sections 2.1 and 2.2, you must

- Use the MNB model and the 5-fold cross validation approach to determine the *classification accuracy* and the *training* and *test time* based on a set of 10,000 documents in the 20NG dataset, which can be downloaded from http://students.cs.byu.edu/˜cs453ta/ projs/20NG.rar, without applying feature selection.
- Use the trained MNB and the 10,000 documents in 20NG to determine the effects of its *accuracy*, as well as its *training* and *test time*, when considering different vocabulary sizes. In accomplishing this task, use the 5-fold cross validation approach and for each one of the four vocabulary size values M ($\in$ { 6200, 12400, 18600, 24800 }) (i) perform *feature selection*, (ii) determine the *classification accuracy*, and (iii) determine the *training* and *test time*.

In the 5-fold cross validation approach, each experiment is repeated 5 times, i.e., each time a different subset of 20NG is used for the training and testing purpose. Thereafter, the averaged classification accuracy and the training and test time of each iteration should be reported.

4 Grading Criteria

The assignment is worth 200 points, and the breakdown of the point distribution is given below.

- Implementing the *Multinomial Naïve Bayes Classifier* is worth 120 points.

- Implementing *Information Gain* for feature selection is worth 30 points.
- A *detailed report* which discusses the various implementation choices (such as the size of the vocabulary or data structures used to enhance the performance of the MNB) and experimental results on MNB is worth 50 points. To create a complete report, you are required to include (i) a *diagram* which shows the *accuracy* of the MNB classifier with and without applying feature selection based on *IG* and a *discussion* on the performance of the MNB in terms of accuracy based on the selected features, (ii) a *diagram* which shows the *training* and *test time* using different vocabulary sizes and a *discussion* on the training and test time of the MNB, and (iii) an overall *conclusion* using the trained MNB for document classification based on the choice of vocabulary size to achieve the best training and test time without significantly affecting the classification accuracy.

Besides the written report, you are required to email the TA a zip file with your implementation of the MNB.

Hint: In verifying that your MNB implementation is working adequately, it is suggested that instead of using the subset of 10,000 documents in 20NG, you assess the classes and methods implemented for training and testing the MNB model using the documents shown in Slide #15 in the Lecture Notes of Chapter 9, which should significantly reduce the time spent in debugging the implementation of the MNB.