Machine Problem 1 of CS 165A (Winter 2019)

University of California, Santa Barbara

Assigned on January 22, 2019 (Tuesday)
Due at 11:59 pm on February 19, 2019 (Tuesday)

Notes:

- Be sure to read "Policy on Academic Integrity" on the course syllabus.
- Any updates or correction will be posted on the course announcements page and piazza, so check them occasionally.
- You must do your own work independently.
- You are required to implement all the details of these tasks by yourself, so you cannot invoke other existing codes or tools. However, some mathematical tools, such as numpy and scipy, are fine.
- Python is the only programming language supported by TAs, so you'd better use it.

---

1 Motivation

After people purchase some goods, they usually write some reviews about them which may be positive or negative. It is an interesting problem that if we can train a classifier to identify whether a review is positive or negative. It is called Sentiment Classification in Natural Language Processing (NLP) research field, which is quite useful in industries. Starting from the raw review text data, in this project, we will do a sentiment classification task.

2 Dataset

You can download the dataset from the course page. Since we are interested in positive/negative review classification, this is a binary classification problem. You'll find the dataset contain four files:

1. `training_pos.txt`
2. `training_neg.txt`
3. `test_pos_public.txt`
4. `test_neg_public.txt`

The first two files are the training set, containing positive and negative reviews, respectively. The rest two files are the test set, also containing positive and negative reviews. Each line of a file is a review, which can be viewed as an instance in the classification task. The "public" means you have access to partial test set where you can evaluate the performance of your classifiers. Besides, we have "private" test set:

1. `test_pos_private.txt`
2. `test_neg_private.txt`

which are reserved for us to evaluate your work and will not be available.

3 Task Description

3.1 Extracting Features

First of all, after you download the raw data, you need to implement feature extraction to make them to be numerical data. There are two kind of features that you should implement:

1. Term Frequency-Inverse Document Frequency (TF-IDF)
2. Bag-of-Words (BoW)

You may learn how to extract these features by yourself.

Please keep in mind you can NOT extract features of training set and (public) test set simultaneously. You may build a vocabulary list of training set first, based on which you can extract features of training set and (public) test set. Also, in your implementation, you should prepare to extract features of (private) test set. Though it is not available, the feature process is the same as (public) test set.

After you build the vocabulary list from the training set, you may find there are some new words in the (public) test set, which will happen to the (private) test set as well. Just ignore them and still use your vocabulary list to extract features.

Optional Task: Removing Stop Words (Bonus 10 points)

After downloading the raw review data, if you take a look at them, you may find there are lots of common words, such as “a”, “and”, “they”, and so on. In fact, they are called stop words in NLP, which have little to do with the key task, thus you’d better remove them before you extract features. Unfortunately, there is no single universal list of stop words. You may find a list by yourself. Anyone who completes this optional task will get bonus 10 points.

3.2 Implementing and Training Classifiers

When you get the numerical data (features of the data), you need to implement two kinds of classifiers:

1. Gaussian Naive Bayes classifier
2. Multinomial Naive Bayes classifier

After you get your classifiers, you may train them on the training set and then evaluate them on the (public) test set. You should write down the prediction accuracy that you get.

3.3 Writing a Report

Finally, you are required to write to a report to describe all the details of your implementation. This report must be in PDF format, which can be generated by Microsoft Word or LaTeX. A PDF picture of a handwritten report will not be accepted.

First, you may write how did you extract features and implement the classifiers. If you did remove the stop words, of course you should report it.

Then, you need to report four kinds of prediction accuracy on (public) test set from your program:

1. Accuracy of Gaussian Naive Bayes classier using TF-IDF feature
2. Accuracy of Gaussian Naive Bayes classier using BoW feature
3. Accuracy of Multinomial Naive Bayes classifier using TF-IDF feature
4. Accuracy of Multinomial Naive Bayes classifier using BoW feature

4 Submission Guidance

Use the CSIL `turnin` command to submit the necessary files and directories. Note that all directory and file names are case sensitive. For this assignment you need to use the following command:

```
% turnin mp1@cs165a mp1
```

Once it's successfully turned in, you will see a confirmation message on screen like:

```
*** Turnin OF mp1 TO cs165a COMPLETE! ***
```

Otherwise, please check your network or other possible issues.

For exchange students that do not have a CSIL account:

1. Obtain a Proof of Registration to the class, usually with a Receipt from Extension.
2. Go to HFH 1140E with the Proof of Registration then someone at the help desk will get the account process started.

Once you have finished developing your code, copy all necessary files into a directory named "mp1". The grader will execute your programs. So your code must run without reference any external files or libraries that are not included in your submission.

We highly recommend you to use Python for this programming assignment. For other programming languages, please include steps about how to run your programs in the report.

You may use any OS/compiler version for development, but **your final code must compile and run on the CSIL machines**. Make sure you compile and run the final version of your code on CSIL machines before turning it in. The executable of your program must be named "NaiveBayesClassifier".

For grading, we will use the same training dataset but (private) test set. So after your implementation, make sure your program is able to read the following four files. For example, "NaiveBayesClassifier.py" for Python. And your program will be called with 4 command line inputs like:

```
   python NaiveBayesClassifer.py training_pos.txt training_neg.txt
test_pos_private.txt test_neg_private.txt
```

5 Grading Policy

We will evaluate your submission according to both implementation and report. If plagiarism is identified, no scores will be given to this assignment and your department will be notified.

For implementation, we are interested in its efficiency, performance, and code style.

For report, we are interested in whether you can concisely and logically explain all the components you used in your implementation.

The total points of this project is 100 points, which are distributed as:

1. TF-IDF feature extraction (15 points)
2. BoW feature extraction (15 points)
3. Gaussian Naive Bayes classifier implementation (15 points)
4. Multinomial Naive Bayes classifier implementation (15 points)
5. Performance of your classifiers based on your extracted features (30 points)
6. Report (10 points)
7. Removing stop words (**10 bonus points**)
8. Also, we encourage you to extract best features and implement best classifiers as possible as you can. When we get your programs, we will test them and sort them by the sum of prediction accuracies. Anyone in the top three will get **30 bonus points**. And **20 bonus points** for the fourth and fifth, **10 bonus points** for the sixth to tenth.