CS165A
Machine Problem 1

Deadline and Submission

- Thursday, February 18, 2015, 11:59pm
- Submit via the **turnin** command on CSIL workstations.
- Submit all the necessary code files, makefile if applicable, and a two page report.
- Write your code on your own!

Task

- Write a program that can classify product reviews into *Positive* or *Negative*.
- Train a *Naive Bayes Classifier* using a training dataset and apply it on a testing dataset (supervised learning).

Naive Bayes Classifier

- The Naive Bayes classifier should calculate the probability that a new product review (unlabeled document) is *Positive*, the probability that it is *Negative,* and choose the most probable.

- As the name implies, this type of classifier uses the Bayes Theorem formula to calculate the above probabilities.

Example

- A new product review is submitted and among other words it contains the word “broke”.
- What is the likelihood that the review is *Negative*?
- P(Class=Negative|word=broke) = ?

Example cont.

P(Class=Negative|word="broke") =

*P(word="broke"|Class=Negative) * P(Class=Negative) / P(word="broke")*

How do we train the classifier to learn these probabilities?

Choose your Features

- Choose single words (e.g. “broke”, “bad”, “love”) as the possible features of your classifiers is the basic approach.
- However:
  - Are all words equally important? “exciting” vs. “the”
  - Do words occur independently? e.g. “like” vs. “not like”

Technical

- Write your program in either **C, C++, Java, or Python**.
- Make sure your code executes properly on **CSIL**. This is where we will grade your project!
- **You can’t use third party libraries, modules, or frameworks.** Only native libraries available on CSIL are allowed.

Program Input

- When your program is executed you need to be able to pass **two command line arguments** which are the file names that contain the training and testing datasets:

- ./NaiveBayesClassifiers testing.txt training.txt

- Then your program can open these two files to load the datasets. <u>Do not hardcode the file names in your code.</u>

Program Input cont.

- The two dataset files have the same format. Every separate line contains a single product review.

- Every line begins with a single character that indicate the product’s class (0 for Negative and 1 for Positive) followed by a *tab* character “\t”.

- The rest of the line contains the text of the product review. Product reviews will contain no new line characters. Product reviews will contain multiple sentences.

Program Output

- Be very careful to follow the correct output format guidelines because output will be parsed by a grading program.

- Your program must output all the extracted labels for the training dataset, one per line. **Print only a single characters, 0 or 1**, followed by the new line character.

- In the end, you need to print 4 more lines:

  - The time it took to train your classifier in seconds
  - The time it took to run your classifier on the testing dataset.
  - The accuracy of your classifiers on the training dataset.
  - The accuracy of your classifiers on the testing dataset.

Program Output Example

```
0
0
1
1
0
1
0
0
1
5 seconds (training)
17 seconds (labeling)
0.913 (training)
0.743 (testing)
```

How many documents exist in the testing dataset?

Program Output

- To test if your program’s output is formatted correctly you should use the provided jar.
    - If you don’t have java installed on your own machine, you can run this program on CSIL.

Running Time and Accuracy

- Make sure that your program will terminate within 15 minutes during grading. Keep in mind that we will use a testing dataset that is x10 larger than the one provided to you.

- Your grade, assuming your program finished in 15 minutes, will be graded mainly based on its accuracy.

- The best 3 projects in terms of accuracy will get bonus credit!

Good luck!

Forward any questions to the mailing list
or teogeorgiou@cs.ucsb.edu