CS311 – Spring 2022

HW Assignment #4: Wednesday, April 27th, 2022 at 11:00am

Rules for group work:

I suggest working in a GROUP on this assignment. You may work in groups of 2 students. You will submit only ONE assignment for the group. You may not "divide and conquer" this assignment. That is, all students must be present (virtually is ok) and contributing while any work on the project is being done. In other words, you must "pair-program" to complete the assignment. Along with the honor code, you must include, in a comment at the top of your assignment the following statement: "All group members were present and contributing during all work on this project." This statement will be treated in the same manner as the honor code, so please make sure it is present and true.

For more details on this homework please see the accompanying video (link on canvas).

Part 1: Ethics/Bias in AI

During our next lab session, we will discuss ethical issues in AI. Please read this article about bias in AI and be prepared to reflect on your reading in class on Monday, April 25.

Part 2: Sentiment Analysis with Naïve Bayes

The goal of this assignment is to implement a Naïve Bayes Classifier that analyzes the *sentiment* conveyed in text. When given a selection of text as input, your system will return whether the text is positive or negative. The system will use a corpus of reviews as training and testing data, using the number of stars assigned to each review by its author as the label (1 star is negative, 5 stars is positive).

The Training Data

The first step of building the classifier is to compute the probabilities required for Bayesian learning. To train your system, you will be using data from reviews collected from a website called 'Rate it all' (which shut down late last year). The reviews come from a variety of domains, e.g., movies, music, books, and politicians, and are stored in individual text files where the file name is of the form *domain-number_of_stars-id.txt.* For example, the file "movies-1-32.txt" contains a movie review that was assigned one star by its author, and has an id of 32. Please note that the id number will be of no use to you for this assignment. Take a moment to explore and get familiarized with the format of the data. The reviews you should use for training are stored in the 'train' zip file.

For your system, you need only use reviews that had one or five stars, where the reviews with one star are the "negative" training data and the reviews with five stars are the "positive" training data. Zip files of the training (and testing) data are located on canvas. You will have to unzip the files to use them.

<u>**Provided Code**</u>:

I have also provided you a skeleton file SentAnalysis.java that contains a method `readFiles()`. It takes as parameter a String, which should be the name of (or path to) a folder of files and returns an ArrayList (of Strings) containing names of all the files in the given folder.

This program can be run in one of two modes: evaluation and interactive. The interactive mode should allow a user to enter text via the prompt and then classify the text as either positive or negative. The evaluation mode asks the user for the name of a folder of documents to classify and it should then classify the documents (only documents rated as 1 or 5) in this folder as either positive or negative and print out some performance statistics on the results (see below for more details on these statistics). To run the interactive mode, simply run the program without any command line arguments. To run the evaluation mode, run it with the command line argument "evaluate". **Take a moment to explore the code provided in the skeleton file so that you don't create extra work for yourself.**

You are welcome to change the method headers and return types of the given methods but you must have these separate methods that complete the specified tasks.

<u>**Train the System**</u>

Now that you're familiar with the training data, your next task is to train the classifier. Given a **target document** (a document to be classified), Naïve Bayes Classifiers calculate the probability of each **feature** (i.e. unique individual word) in the target document occurring in a document in each **document class** (i.e. positive or negative) to find the class that is most probable. To make these calculations, the system must store the number of times that each feature occurs in each document class in the training data. For example, the word "good," may occur 1500 times in the positive documents, but only 200 times in the negative documents.

You will training your classifier in the `train()` method. You should use a dictionary i.e. hash table (HashMap in Java) to store all of the counts. In particular, you'll have two dictionaries: one will hold all of the words that occur in positive documents and the frequencies at which they occurred, and the other will do the same for the negative documents.

Once you have created the dictionaries, it is time to train the system.
Write the `train()` method that trains the system by doing the following:

- For each file name in the training folder, parse the file name and determine if it is a positive or negative review; update the corresponding count.
- For each word in the file, update the frequency for that word in the appropriate dictionary.

__Start Classifying!__

At this point, you now have all the data you need stored in memory, and can start classifying text! Given a piece of text, your `classify()` method should output its correct document class (i.e. positive or negative). To do this, you will implement a Naïve Bayes classifier.

<u>Here is a summary of what we discussed in lecture about Naïve Bayes Classifiers</u>:

Naïve Bayes Classifiers find the conditional probability of a document *d* being a member of a class *c* as follows. Assume we are trying to calculate the probability that a target document *d* is positive given the set *f* of *n* features of *d.* Then the probability that *d* is positive given *f* is:

$$\Pr(positive\,|f) = \alpha \Pr(positive) * \prod_{i=1}^{n} \Pr(f_i \mid positive)$$

The first probability on the right side of the equation is the probability of any document being positive:

Pr(*positive*) = #positive reviews / total # of reviews

The second term on the right side of the equation is the product of the conditional probabilities of each of the features occurring given that a document is positive. For example, suppose the text *d* is "I love school". Then the above equation would be computed as:

$$\Pr(positive\,|f) = \Pr(positive) * \Pr("I" \mid positive) * \Pr("love" \mid positive) * \Pr("school" \mid positive)$$

where:

Pr($word_i$ | positive) = (# $word_i$ in positive docs) / (# unique words in all positive docs)

When training the system, instead of storing these probabilities, just store the frequencies needed to calculate the probabilities on the fly. In particular you will need to store:

- how often each word appears in the positive documents
- how often each word appears in the negative documents
- how many positive documents there are
- how many negative documents there are
- total number of documents

For outputting the result of classification, we will use the Maximum Likelihood approach. Essentially, this means that you calculate the probability of the text being in each class and output the class corresponding to the highest probability. More specifically, you will calculate:

$$\Pr(positive\,|f) = \Pr(positive) * \prod_{i=1}^{n} \Pr(f_i \mid positive)$$

$$\Pr(negative \mid f) = \Pr(negative) * \prod_{i=1}^{n} \Pr(f_i \mid negative)$$

And output positive if Pr(*positive* | *f*) was larger or negative if Pr(*negative* | *f*) was larger.

You will find that two problems will arise in building this basic classifier. I suggest that you build the classifier first, and then address the problems, which are:

1. Smoothing: This problem is subtler than the other problem (see below) as it won't produce a bug, but will throw off your calculations. Suppose the word "yuck" never appears in a negative message during the training. Then the classifier would assume Pr("yuck"|negative) = 0 which will cause the entire Bayesian formula to be zeroed out and therefore the message would not be classified as negative. Clearly, a single word should not be given that much power in our decisions.

   To address this issue, we use **Smoothing**. Essentially, we assign a very small probability to words that do not appear in the training. To apply smoothing we compute Pr(word | negative) as follows:

   $$\Pr(word \mid negative) = \frac{N(word, negative) + \lambda}{N(\#\ words\_in\_negative\_docs) + n\lambda}$$

   where:
   $N(word, negative)$ is the number of (non-unique) occurrences of *word* in negative reviews
   λ is a small value (start with 0.0001 and experiment with this value)
   $N(\#\ words\_in\_negative\_docs)$ is the number of unique words in negative documents
   n is the number of features (i.e. unique words)

2. Underflow: Although smoothing will prevent the probabilities from being zeroed out, our computations may still suffer from an underflow error, which occurs when we multiply several really small floating-point numbers. The resulting value may be too small for a floating point variable to encode, and will therefore be set to 0.

   Solution for Underflow error: Log Probabilities

   The standard solution to this problem is to calculate the sum of the logs of the probabilities, as opposed to the product of the probabilities. This solution is referred to as **log probabilities**.

   We would compute the equation below:

$$\Pr(positive \mid f) = \Pr(positive) * \prod_{i=1}^{n} \Pr(f_i \mid positive)$$

as follows:

$$= \log(Pr(positive)) + \log(Pr(f_1|positive)) + \log\big(Pr(f_2|positive)\big) + \cdots + \log(Pr(f_n|positive))$$

The logs above are in base 2. In your classify method you should compute the last equation (i.e. the summation) as it will result in quicker code (since adding is quicker than multiplying) and avoid underflow.
Note that if we applied the log probabilities technique before smoothing, then we would still have a problem – if for some f_i, $\Pr(f_i \mid positive) = 0$, then we would try to compute log(0) which would result in an error.

If you've followed the instructions above, you should now have a classifier that can be used as follows:

```
Text to classify>> I love my AI class!
Result: positive
```

Reflection Question: Take a moment to celebrate your success. Try out your classifier on a few sentences and see when it performs well and when it performs poorly. What are three examples of sentences (or any text) on which you think your classifier failed? For each example, why do you think it failed? What was difficult about the target text?

Improve your Classifier!

Congratulations! Once you reach this point, you will have built your first Naïve Bayes Classifier! In the previous part, you probably realized some ways to improve your system. In this part, I would like you to be creative and implement these ideas and build *your best classifier*. Time permitting, we'll hold a competition in class to see whose classifier does best on a set of documents that I'll choose randomly. The outcome of this competition will not have an impact on your grade. For this part, make a copy of your Java file and add the word "Best" to the end of the filename i.e. SentAnalysisBest.java. This will make it easier for us to grade these parts separately.

The most important research problem with respect to Naïve Bayes Classifiers is the choice of features used in classification. Thus far, you've used only "unigrams" (i.e. single words) as features. Another common feature is "bigrams" (i.e. two-word phrases). Another feature might be the length of the document, or amount of capitalization or punctuation used. You may experiment with these ideas for your *best classifier*. **For *your best classifier,* you must include at least one other feature in your classification**.

<u>Part 3: Evaluate Your System</u>

Now that you have a working system, you can evaluate how well your system performs. You will do this in the `evaluate()` method. To evaluate, you should test on the files in the 'test' zip file (never train and test on the same data). We'll consider two different evaluation measures:

• Accuracy: number correctly classified / total classified

• Precision: number correctly classified as positive / total classified as positive
number correctly classified as negative / total classified as negative

<u>Clearly label (with comments) how/where you are keeping track of these counts and calculating the final results</u>. In particular, you should have variables for each of these counts (e.g. numcorrect, total, numpos, numneg, etc.) and increment them when appropriate.

Using these outputs, I'd like you to evaluate how well your *best* system works. In particular, I'd like you to compare your base system (that you completed in part 2) to your *best* classifier (that you completed in part 4). In your **ReadMe**, please discuss how the two systems compare based on their accuracy and precision. Also reflect on why you think the systems performed well (or poorly). Describe some ways that you could improve the system even more.

As a point of reference, the standard classifier should have the following results:
Accuracy: 76.7%
Precision (positive): 76.5%
Precision (negative): 76.9%

And an improved classifier should get something as follows:

Accuracy: 83.9%
Precision (positive): 81.0%
Precision (negative): 87.2%

<u>Output</u>:

We would like to grade your programs as efficiently as possible. So please make sure your program does not output anything other than what is asked for above. Points will be deducted for extraneous output/debugging statements.

<u>ReadMe</u>:

As always, please provide a ReadMe that includes the following:

1. The names of the students in your group.

2. Names of files required to run your program.
3. Any known bugs in your program (you will be less penalized for bugs that you identify than for bugs that we identify).
4. The discussion of results and data asked for in Part2 and Part3.

Submission:

Please include, in a comment at the top of every file you submit, the following:

- Full names of students in the group
- Group honor code: "All group members were present and contributing during all work on this project"
- Acknowledgement of any outside sources of help (discussions with other students, online resources, etc.)
- Middlebury honor code

Zip the following files and use the HW4 link on canvas to submit exactly **ONE** zipped file per group:

- SentAnalysis.java – Your base classifier (from Step 3)
- SentAnalysisBest.java – Your best classifier (from Step 4)
- ReadMe

Name your zip file as follows: HW4_<last names of all group members>. So, for example, if your group consists of the last names Alvarez and Baker, you should name your file HW4_AlvarezBaker.

Grading

[100] Program:
- [25] Part 2: Training
- [25] Part 3: Base classifier
- [20] Part 4: Best classifier
- [20] Part 5: Evaluation/discussion
- [10] Code Comments

This assignment was adapted from Sarah Owsley Sood at Northwestern University.

Extra Credit

There are two ways to earn extra credit for this assignment.

1. Build the best classifier. We will have a class competition to see which team can build the most successful classifier in terms of accuracy and precision (see above). The winning team will earn extra credit (+10 points).

2. Add a *neutral* label. Try to modify your code so that it can also classify text/documents as *neutral*. If you decide to do this extra credit, please submit a separate driver class, SentAnalysisNeutral.java, in addition to the classes specified above (+10 points).

Part 3: Written Problems

1. Bayesian Inference

Your Doritos are gone and you have two roommates as suspects – Alice and Bob. You know the following:

- In previous Cheeto thefts Alice was the guilty robber 85% of the time and Bob was guilty only 15% of the time.
- Your neighbor believes she saw Alice eating Cheetos yesterday, but she has poor eyesight. You estimate that her rate of Alice/Bob differentiation is 80%, so 80% of the time that she thinks she sees Alice, it's actually Alice, and 20% of the time it's actually Bob. Similarly, if she thinks she sees Bob, she is correct 80% of the time.

a) What is the probability that Alice stole the Cheetos? Show all work for full credit.

b) If your neighbor thought she saw Bob, what would be the probability that Alice stole the Cheetos? Show all work for full credit.

2. Bayesian Networks

You have a bag of three biased coins *a*, *b*, and *c* with probabilities of coming up heads of .2, .6, and .8, respectively. Each coin is equally likely to be drawn from the bag. You draw one coin randomly from the bag and flip the coin three times to generate the outcomes X_1, X_2, and X_3.

a) Draw the Bayesian network corresponding to this situation and give the values in the conditional probability tables (CPTs).

b) For each coin, calculate the probability that the coin was drawn if the observed flips came out to heads twice and tails once (not necessarily in that order). ? Show all work for full credit.