John Rehbein

MP1 Report

• Architecture: Explanation of your code architecture (briefly describe the classes and basic functionality)

I used the strategy design pattern for computing conditional probabilities. I made an abstract interface for this. One method that gives you the probability given for a given category and document and another that will train. I used this to create a class that could combine and weight heuristics and this allowed me to minimally change the model class to check out different combinations of Heuristics (or dependent variables). The document class was a basic data structure that contained the category and words in a Document. The NBC model class was pretty basic and could probably be reused for other programs since it was abstract enough.

• Preprocessing: How you cleaned and how you represent a movie review document (features)

I found a magical function from the internet that made parsing the documents a bit easier. The tools for doing this kind of thing in C++ are pretty nasty to work with, and the function I found might create a memory leak which is kind of bad, but I figure it would be easy to fix if I needed to. I delegated processing the documents to the document class so I separated the text documents into words there. I did this so I would not have to repeat this process on the model side of the project.

• Model Building: How you train the Naive Bayes Classifier

I used the bag of words model talked about in lecture. But I also decided to be clever and make my own heuristic. I made a heuristic that measured the probability of a category given two words that were within n words of each other. This took a lot of time to process, was decently tricky to program, and I was really optimistic, but it turned out to perform the same as the bag of words model except that it became super overfitted to the training dataset. I got about 0.99 accuracy on the training dataset using this heuristic.

• Results: Your results on the provided datasets(accuracy, running time). Also give a sample of the 10 most important features for each class (positive or negative)

Using just the bag of words model the best accuracy I got on the testing dataset was 0.844. The running time was “0 seconds” but when I used some of the more processing intensive features I made I got like 11 to 30 seconds in running time depending on how I set the parameters.

The 10 most important features that weren’t things like pronouns and other useless words were
Features: Movie, Film, One, All, Like, just, about, some, good, more

• Challenges: The challenges you faced and how you solved them

- Weaknesses: Weaknesses in your method (you cannot say the method is without flaws) and suggest ways to overcome them.

The biggest challenge I faced was fighting the csil compiler. Since I developed the project on my mac the compilers were different. Xcode automatically imports certain things and uses special flags so that led to a lot of confusion and debugging. Everything else went smoothly and steadily. Aside from one bug that I am still annoyed about. If you ask for the minimum value of a float in C++ you get the minimum positive value opposed to the minimum negative which is super counterintuitive. Probably the biggest weakness is my code was not as modular as I'd like it to be and I perform a lot of repeat computations. Some of my code, but not that much is super sloppy.