For my first means of testing, I just hardcoded the simple-train-tags/sentences into my program. I wrote down on paper what the emission and transmission keys and values should be for the simple-test-sentences, and then printed out the emission map, transmission map, and tags as I coded, comparing the maps to my own calculations and comparing my tag accuracy to the solution tag accuracy listed in the assignment. I ended up finding an error in my transition map this way, because I was getting positive values in my transmission map that shouldn’t have been there post-log. Once I got the 32 correct words and 5 incorrect words that the assignment description said I should have, I wrote a method to take user input so I could check on sentences of my own. The accuracy definitely dropped overall (unless I only created sentences using words from the simple-test-sentences) because my code just didn’t recognize the words. When I retrained my Viterbi using the brown-train-sentences/tags instead, however, the accuracy for my user-input tags was significantly better. Understandably, my program didn’t know what to do when it read my name (which it mis-tagged), but other than that it was quite accurate on what I would consider normal English sentences + numbers. Even using sentences with homonyms, I feel like the program did a lot better than I expected, which was exciting. I attached a screenshot of the sentences with their tags; of the 6 words I tested in sentences (each used as two different parts of speech) it got 11/12 of those tags correct ☺ The word that was tagged incorrectly was “fair,” used once as an adjective and once as a noun, but tagged as an adjective both times. For my next phase of testing, I wrote the code to get the exact percentages of correct tags for files. It runs through each sentence of test code and each sentence of test tags, uses my Viterbi to get tags for the sentences, and compares the tags, printing out the number of correct and incorrect tags, and the percent of tags that were correct. For my simple-test files, I had an accuracy of 86.5 percent, with 32 words correct and 5 incorrect (verifying the results I got when I went through and checked the tags by hand), and for my brown files I had an accuracy of 96.5 percent, with 35109 correct and 1285 incorrect. I also temporarily added a line to that same test method to print which words were incorrectly tagged because I was curious. For the simple test files, “saw,” “to,” and “bark” were mistyped (“to” and “bark” twice each). The list of incorrect words for the brown file was much longer, but just skimming through it, I noticed that a lot of the mis-tagged words were words that started with dollar signs, abnormal decimal numbers or fractions in general, acronyms, and names. Which makes sense, because the less often a word appears, the less accurate the training will be, and then the tagging will be messed up. Interestingly, although the accuracy goes down for unseen penalty values around -10, it does not go up when the penalty becomes more negative. -10000 has the same accuracy as -100. The ideal penalty appears to be in the range between -17 and -19; at this value for the brown file, four more words are labeled correctly as opposed to when -100 was used.