Nick Largey
Text-Mining
Assignment 2

Task 3:

Unigram:

Confusion Matrix:

	Blues	Rock	Country	Metal	Pop	Rap
Blues	0	1	0	1	0	0
Rock	1	1	0	0	0	0
Country	2	0	0	0	0	0
Metal	2	0	1	0	0	0
Pop	1	1	0	0	1	0
Rock	2	0	0	0	1	0

F1 Scores: {'Blues': 0, 'Rock': 0.4, 'Country': 0, 'Metal': 0, 'Pop': 0.4, 'Rap': 0}

Average F1 Score: 0.13

Bigram:

Confusion Matrix:

	Blues	Rock	Country	Metal	Pop	Rap
Blues	1	0	0	0	1	0
Rock	1	0	0	1	0	0
Country	2	0	0	0	0	0
Metal	2	0	0	1	0	0
Pop	3	0	0	0	0	0
Rock	3	0	0	0	0	0

F1 Scores: {'Blues': 0.14285714285714285, 'Rock': 0, 'Country': 0, 'Metal': 0.4, 'Pop': 0, 'Rap': 0}

Average F1 Score: 0.09

Mixed Model:

Confusion Matrix:

	Blues	Rock	Country	Metal	Pop	Rap
Blues	0	2	0	0	0	0
Rock	1	1	0	0	0	0
Country	2	0	0	0	0	0
Metal	2	0	1	0	0	0
Pop	2	0	0	0	1	0
Rock	3	0	0	0	0	0

F1 Scores: {'Blues': 0, 'Rock': 0.4, 'Country': 0, 'Metal': 0, 'Pop': 0.5, 'Rap': 0}

Average F1 Score: 0.15

Task 4:

In the attached text files, I show some of the output of the values for different areas of EDA I did. I began by getting all of the tokens and their associated probability (tokens.txt) so that I could go through and see if there was anything that could be done to clean the data more so that it would be representative of each genre. I believe that with the the regex's to sub newlines with a space, split the words from the test set that were line breaks but then concatenated together, and remove all punctuation from the training and test data did as best a job as I could do with this model architecture as far a data cleaning goes.

I then wrote out the probabilities of the tokens for both the uni-gram and bi-gram models (tokens.txt) so that I could explore how the probability of each token was presenting itself. I used a few different methods to try and figure out the best way to calculate this, but since my math background isn't as strong as it should be, it was mainly trial and error. But the token probabilities I landed on provided the best results for all 3 models.

In order to find the best lambda for the Mixed Model (prob.txt), I had a variable that was set whenever the best lambda value was found by keeping track of the Current True Positive count and comparing it against the Best True Positive count, if it was greater, the best lambda was set. Originally I had this set to decrement the lambda value by 0.1, but found that 0.01 was more accurate and showed interesting clustering results for lambda's that would return the same amount of True Positives. So for my model, it seems that any lambda from 0.8 to 1 would return the max True Positives.

Task 5:

Overall, each model was a spectacular failure. The F1 score of 0.15 for my Mixed Model was the best I was able to achieve during all of my testing. I think that one of the biggest issues is that the size of the training and testing data is just too small. There were ~6500 tokens total for the uni-gram and ~10,000 for the bi-gram for all of the genres, a good data set would probably have somewhere close to a thousand times the total size of the training tokens for each genre. There's also only roughly 130 songs used to train, which is not a good representative sample size.

To be honest though, I was expecting somewhat better results. At the end all 3 of my models were heavily favoring "Blues" as the category, which in some historical context would make sense,

since we are looking at American music, and the Blues is very much the basis for all of these genres when traced through-out time. Blues also happened to have the least amount of tokens, but this should have less impact by applying the Add-1 Laplace smoothing in the freq_to_prob() functions in each model. Though, further refinement or a different approach would be needed to actually produce results that have and significant merit.

I was a bit surprised to see that the uni-gram out performed the bi-gram model as well. Even though the results vary wildly from the Bi-gram's. For example, Song #14 of the test set gave the results:

Song # 14
Max Probability Genre: Rock with probability -148.79784913217398
True Genre: Rock
OrderedDict([('Blues', -172.12391957910876),
('Country', -195.9395571953843),
('Metal', -195.82455476300524),
('Pop', -177.6985088804608),
('Rap', -178.6316665226306),
('Rock', -148.79784913217398)])

which was correctly predicted, but the variance in the probabilities is 47.14 points which is about a 30% difference in best to worst probabilities.

And for the bi-gram model the results are:

Song # 14
Max Probability Genre: Metal with probability -1.2135092336352065
True Genre: Rock
OrderedDict([('Blues', -17.599313891387975),
('Country', -26.601212493266736),
('Metal', -1.2135092336352065),
('Pop', -15.14893211605699),
('Rap', -53.18633847667919),
('Rock', -33.812203147215534)])

Which has a difference of 51.97 points, which is close to the uni-gram, but it's about a 440% difference here, and it didn't get the prediction correct!