Senior Project First Semester Write-up

At the start of the semester we pitched around various ideas involving text analysis. After a little while, we decided an interesting direction to go would be to try applying the concepts of text analysis to music (it can be viewed as a language of sorts).

I had a very basic understanding of midi file formats from using some song writing software previously, so I figured their structure would lend itself to this project well, since they (at least in some formats) are already broken up by parts, and are written as a series of events representing starting and stopping pitches at various times. So after playing around briefly with trying to decode a midi file’s structure in hex (using marry had a little lamb as an example), we decided to instead use jmusic (as suggested by Dr. Sigman). jmusic is essentially a java library built to read in, write, and perform some analysis on midi files, so the switch to it meant we wouldn’t have to hand code our own utility to change a midi file into a readable series of notes with values for pitch and rhythm, and instead jump right into analysis on a song.

Ultimately we decided to look at trying to do genera identification on a song. We felt that looking at the four musical periods of western music would be a good route to go. So for the first portion of the semester, I worked on building a system to perform unigram analysis on a song looking at two things, the rhythmic values of notes and the interval shifts from one pitch to the next. As for storing these values after reading them, during execution a balanced binary tree is used so that it is quick to find previous values for updating them, and storing the final results was done as just a text file dump of the finished tree. After this was functional, I set it up to read in the text file if it existed already so that a library could be built from multiple songs. For initial testing with this, I just hard coded the file paths, later setting up a method for just running it on all songs in a folder.

After we had a library and looked through the values we were getting back, we decided some of them on the extremes of the spectrum for pitch were probably unassay, as they’d only have happened a few times, so we decided to concatenate it down to just -24 to 24 (a two octave jump in either direction). As for the rhythm values, the results from Chopin songs were returning very long float values and consequently, notes that were likely intended to be the same rhythmic value were coming back as separate, we fixed this by having it round to two decimal places for the rhythm value.

When we were satisfied that we had decent looking libraries (one for Bach and Chopin respectively), we looked at how to best do analysis with them. After working out the probability theory we would use, it quickly became apparent that underflow and overflow would be very likely for the number of notes in a full song (as the likelihood was the percentage of occurrence raised to the number of occurrences). To combat this issue, we initially just did analysis on a phrase from a song. But we ideally wanted to be able to analyze a full song.

In order to accomplish this, I built a class that would run through a song in chunks, and we could ideally use those individual results to weigh the overall likelihood of it being one or the other for the entire song. This method seemed to work after a good amount of fighting through bugs. However, as seen in the case of a Chopin song (raindrop prelude), the result wasn’t always pointing to the correct composer. So at that point we decided to try moving to bigram analysis and try looking a pitch in a different way.

So most recently I worked on altering the system to analyze in bigrams of both rhythm and pitch with scale degrees to keep pitches in a more relative sense to the key. Additionally, I set up the system to function more generally that it was, since it was originally structured around the two composers being used for testing, and later should be usable for 4 composers typically representative of their time period. Where the system is at right now is that it is successfully building libraries on the bigrams, but song analysis has found a new bug since the switch to them. So over break I plan to test and get that to a functional state and then move to just running a good amount of tests to see precisely how successfully it is identifying composers, also expanding the song collection to include more songs and also two artists representative of the other musical periods.

For next semester, we will want to write up the results, so having a large number of tests run before then with the system will give us a much better idea of its reliability at successfully identifying a composer, so that if it does fail to correctly identify a song, then we can see if it’s perhaps just some characteristic of that song, that perhaps it’s a little out of the composers normal styling’s or something similar, if it is correct for most songs in general. If time allows, we’ve also discussed trying to use the library statistics to try to generate a random song in a composer’s styling. It’s likely we’d want to implement some pre-defined rules for it though, at least to give it a more musical start and end, and thus far harmonies haven’t been a part of the analysis, so it would be restricted to a single voice.