< Project Report : Generating Classical Music Melodies >

Names: Jiyoung Chang, Chase Holdener, and John Tobin

Synopsis:
Our code is designed to take a range of song files (in the format of the Orchset song text files), analyze the pattern of the various notes and their durations in those songs, and use this analysis to generate a new song through a Markov Chain approach. The result is a new song file that can be manually entered into a midi file generator to listen to.

Project Goals:

- We formatted the song text files in Orchset to be analyzable.
- We created a dataset such that for all $note_i$ in the range of notes in octaves 0 through 8 in the musical scale, we have a list of $note_i$ and the number of occurrences of every other note that follows $note_i$
- We created a dataset such that for all $note\text{-}duration_i$ in the note-duration list ("1/16", "1/8", "1/4", "1/2", "3/4", "1", "5/4", "3/2", "7/4", "2"), we have a list of $note\text{-}duration_i$ and the number of occurrences of every other note that follows $note\text{-}duration_i$
- We wrote a procedure to use the probabilities stored in note-dataset and note-duration-dataset to generate a new song through a Markov Chain approach.

Data Description:
The dataset consists of 64 songs each with their own file. Each file has a collection of lines, with each line having a timestamp (in centiseconds) and a frequency (in Hz) that is playing at that time. We retrieved this data from the Orchset dataset cited in the Works Cited section.

Sample data file for a song:

Time (cs)	Frequency (Hz)
0	0
0.01	0
0.02	0
0.03	0
…	…
12.42	867.008
12.43	391.995
12.44	391.995
12.45	391.995
12.46	391.995

(We have written an algorithm to reformat these data sets to be in letter note format (A, C, F#, etc.) instead of frequency--based on frequency to note conversions outlined in https://pages.mtu.edu/~suits/notefreqs.html)

Algorithms:

Our project makes use of three algorithms

- The first algorithm is composed of four procedures, and its function is to interpret the information from the dataset and get it ready for us to make new datasets for both note information and beat information separately. The procedures are as follows:
    - frequencies->notes, which changes the frequencies from the file into music notes.
    - clean-notes, which interprets the file information and uses frequencies->notes to convert the frequencies in the file to music notes.
    - notes-in-order, which combines the timestamps into durations that each note lasts for.
    - clean-beat, which changes the durations into music durations or “beats” (i.e. ¼ ½ etc.).
- The second algorithm is composed of seven procedures, and its function is to take the data that was interpreted by the first algorithm and create two datasets (one for notes, and one for durations or “beats”) containing each possible value$_i$ along with all of the values and their respective number of occurrences that ever followed value$_i$. These two datasets serve to provide the probability information about the chosen song(s) that is used by the third algorithm to predict subsequent notes/durations for each note/duration given. The procedures in the second algorithm are as follows:
    - analyze-pattern-notes, which gives all of the notes that follow a given note along with the number of occurrences of each note.
    - analyze-pattern-beat, which gives all of the beats that follow a given beat along with the number of occurrences of each beat.
    - make-note-list, which produces a list of all possible notes and applies analyze-pattern-notes to each element in order to determine all notes played in the song, every note that followed those notes, and the amount of times each note followed the note for a given song.
    - make-beat-list, which produces a list of all possible beats and applies analyze-pattern-beat to each element in order to determine all beats played in the song, every beat that followed those beats, and the amount of times each beat followed the note for a given song.
    - get-song-files, which allows for a user to choose multiple songs for use in generating the new song
    - make-note-dataset, which makes a dataset of all notes and the notes that follow each note across all songs.
    - make-beat-dataset, which makes a dataset of all beats and the beats that follow each note across all songs.

- The third and final algorithm is composed of two procedures. This algorithm is the actual Markov chain that gives the final output of our procedure. The algorithm works by first generating a random starting note and a random starting duration. Then, using the datasets that were generated by the second algorithm, the Markov chain determines the next note and duration using the probabilities established in the datasets. (The probability for a note or duration to appear after a value_i is determined by dividing the number of occurrences of that note/duration following value_i by the total number of notes/durations that ever follow the value_i) The note and duration are then combined into a single list that is one of the nested lists in the final output. This process then repeats with each new note and duration until the durations add up to the length specified by the user. In the end, the output will be a list of lists each containing a note and a duration for that note. Sequencing these notes and durations in a music sequencing program will yield the generated song. The procedures in the third algorithm are as follows
    - generate-note-or-beat, which generates a new note or beat based on the probability established by the given dataset.
    - make-song, which generates a song in the form of a list of lists each containing the note and the duration of that note.

Analysis of Results:

Our algorithm was successful in generating new songs based on the patterns of notes and patterns of note durations from the Orchset songs. The Markov Chain approach for generating the songs provided some interesting results. We noticed that commonly there would be stretches of the generated song that had a pleasant sound and a natural flow, and then at some point the song would be disrupted and sound unnatural (perhaps by an unlikely note/duration being selected by the Markov Chain). Additionally, since our song-generating algorithm uses only the previous note in generating the next note, sequences of notes sometimes sounded forced; however, it is the nature of music that there is no absolute right or wrong sounding note/duration, so many of our songs can only be evaluated subjectively. We also noticed that if we used only a few songs to generate our new song, the generated song sometimes sounded similar to some of the reference songs, and intuitively, as we increased the number of songs in the databases, the new song sounded less like any of the songs in the database in particular. It is also valuable to note that our algorithms were dependent on the music melody extraction research by the team behind the Orchset dataset. Hence, there could potentially be some bias in the songs we generate through our program depending on how accurate the Orchset MIDI files really are. Another minor limitation is that, when all 64 songs are used in generating the datasets, the program can take a substantial amount of time to run (on the order of minutes). All in all, our algorithms functioned as anticipated in generating new music, and we are pleased with the music that our program produced.

Extensions of our code:

- We could revise our code to be more flexible with choosing ranges of songs to use in the make-song procedure (e.g. ability to select multiple ranges of songs to base the generated song on, instead of 1)
- We could add a “key” parameter to our algorithm to limit the notes that can be selected to be notes of a particular key (to make the melody sound more natural)
- We could make the Markov Chain approach consider the past 3 notes or more, instead of 1, so the sequences of notes sound more natural

Instructions to operate our code:

1. Download the zipped Orchset dataset to a location where the folder can be unzipped, and unzip the folder. (We saved it on the desktop.)
2. Within the "Orchset" folder, there is a "GT" folder. Note the location pathway to get to the "GT" folder on your computer. You will need this pathway later as a parameter for our procedure.
3. Open the "Final_Code_Music_Generator.rkt" Racket file and run the interactions pane.
4. Call the procedure "make-song" with 4 parameters:
   (make-song <song-length> <location-path> <l-bound> <u-bound>)

   song-length = the desired number of measures for the song, entered as an integer (Our code was designed to generate music in the 4/4 time signature at 120 bpm, so each measure will correspond to 2 seconds of song.) Values greater than 1000 can take excessive time to run.
   location-path = the location pathway to the GT folder saved on your computer as a string
   l-bound = the lower bound of the range of songs you want included in generating the database that the new song will be based on, an integer (see note below)
   u-bound = the upper bound of the range of songs you want included in generating the database that the new song will be based on, an integer (see note below)

   NOTE on song ranges:
   The bounds for song ranges can span from 0-63. 0 marks the first song listed in the order provided in the Orchset file, 63 marks the last song in the Orchset file. If you open the "GT" folder

in Orchset you can see this ordering of songs that our procedure is based on. Depending on which values you choose for l-bound and u-bound you can generate music based on specific composers/ groups of composers. The documentation of the "make-song" procedure provides more info.

5. Yay you've generated a song (hopefully)! Now, if you want to hear your masterpiece, you can go to a midi file song generator website and enter the notes with their respective durations. Start at the top of the outputted list, and work your way down entering the notes into the midi generator. https://onlinesequencer.net/ is a good website to try. Jam out!

Works Cited

Bosch, J., Marxer, R., Gomez, E., “Evaluation and Combination of Pitch Estimation Methods for Melody Extraction in Symphonic Classical Music”, Journal of New Music Research (2016)

Michigan Tech University, Frequencies for equal-tempered scale, A4 = 440 Hz
https://pages.mtu.edu/~suits/notefreqs.html

Jacob Morgan and George Burdell, Online Sequencer - Make Music Online
https://onlinesequencer.net/