Milestone 3 Report

Current State of the Project

Milestone 3:
Our goal was to create a recommender system that takes in a wav file of a song, and outputs identifying information about another song in the same genre, such as song title and artist. This was implemented as follows:

Input	A wav file of a song
Processing	Transform the input into a series of spectrograms and classify their genres (using the genre classifier from Milestone 2) before finally taking the average of the outputs for a final verdict on the input's genre.
Output	Song title and artist name of another song in the same genre.

Details:
To accomplish this, we also modified our genre classifier to be trained on spectrograms, rather than song feature arrays reshaped and padded to be valid VGG-19 inputs. Some benefits brought on by this change include:

- Spectrograms, being images, are better suited for classification by the pretrained VGG-19 from Torchvision.
- We can get more data for the more sparsely represented genres in the Million Song Dataset. E.g. by downloading audio samples from Youtube and generating spectrograms from them.
- Spectrograms are a better use of space, since they require less/no padding.

The best genre classification accuracy that we have achieved so far is 70%, with four genres. While this is not the originally planned eight genres, we decided to go forward with this for now with the interest of completing an end-to-end product in time for Milestone 4's deadline.

Going Forward:
We are thinking of trying the following approaches, when time permits, to improve our genre classification:

- Look into incorporating ML architectures like RNNs, that can make predictions based on sequences of data. Right now, we are chopping up our audio samples into 90-second chunks and making 3 spectrograms out of each chunk. Then, each of the three spectrograms from a 90-second chunk become one input channel of three required by VGG-19. A model that could be fed a single spectrogram rather than three channels at a time, and incorporates an RNN (perhaps) before the output layers may take better advantage of the data we have.
- Find how the genre classifier accuracy scales with number of samples per genre, with number of genres held constant.
    - If accuracy goes up with the number of samples per genre, look into getting more samples to improve overall accuracy of genre classification.
- Find how the genre classifier accuracy scales with number of genres, with number of samples per genre held constant.
    - If accuracy increases with decreasing number of genres, Implement an ensemble of genre classifiers to improve overall accuracy of genre classification.

Adjustments to Proposal

- We changed the input to our genre classifier from song features taken from Million Song Dataset, to spectrograms. This is with the aim of improving our genre classifier's accuracy by using data better suited to our VGG-19's image classification powers. (The use of spectrograms was not mentioned in the proposal).
- For Milestone 4 / the final deliverable, we will be removing the need to download files from Google Drive. Instead, data will be stored into saved dictionaries that are packaged with our GUI.

Current Challenges / Bottlenecks

- For improving model accuracy and creating the recommender system, we have turned to using spectrograms. To obtain spectrograms for tracks in the MSD, we attempted to use the 7digital API for acquiring audio samples, but that requires **an API key that we do not currently have access to**.
    - Our alternative was to create a dictionary mapping Youtube links for each track, to genre. Audio samples were then downloaded from these links and associated to the genre they were mapped to.
- The search for some tracks have yielded Youtube videos with irrelevant audio
    - The search for Youtube links was done in the format "<Track Name> by <Artist>" to narrow the scope of results.
    - The URLs collected were further filtered using information from extract_info() function of the youtube_dl library.
- The spectrogram extraction process is slow. E.g. It took 7 hours to generate and store spectrograms for 1000 URLs. This means any changes to our model that requires changes to our dataset that can't be done through our spectrogram dataset class (such as grouping together all the spectrograms from the same song for feeding into an RNN) will be very time consuming to test out.
    - To work around this, the spectrogram extraction/storage loop was designed to be executable in multiple sittings and require as little supervision as possible. Thus, we can work on other tasks while the spectrogram extraction process is at work.
- We have designed our pytorch dataset classes to construct datasets with an equal number of samples per genre. This means one genre with a disproportionately small number of samples impacts the number of samples for all the other genres, and the size of the dataset. E.g. we have 47 samples for Folk music, and therefore, the whole dataset is limited to a size of 8*47.
    - With the shift to using spectrograms, we are now able to get more samples for the most sparsely represented genres in our collection of data.
    - We may also consider simply dropping the worst represented genres from our genre classification entirely.
- The full Million Song Dataset is available at https://aws.amazon.com/datasets/million-song-dataset/
    - This is meant to be hosted on Amazon Drive and getting it onto Google Cloud is a bottleneck we are currently dealing with in order to access the entire Million Song Dataset.
    - **Possible solution:** If we use spectrograms for genre classification rather than tensors with song features from the Million Song Dataset, we will no longer be restricted by the Million Song Subset (10,000 samples), and can get samples for any song in the Million Song Dataset for which we know the genre (~100,000 samples), which will likely improve on the number of samples for which we know the genre and are able to use now (~1000 for 8 genres).

Team Member Contributions

- Carroll, Quinn
    - Worked with team members to add max_loudness feature to model in genreClassifier.ipynb
    - After we switched to using spectrograms for our model, I created class genreClassificationDatasetSpectogram in a paired programming session with Cassiel
    - Worked on Recommender.ipynb in paired programming sessions with all group members
- Jung, Cassiel
    - Collaborated with all team members through zoom on Apr 21st to add one more feature max_loudness. Accuracy did not increase by much.
    - Wrote dataset class with Quinn using zoom screen sharing feature. The class is to get the spectrogram matches with the genre. Has 3 functions __init__, __len__ and __getItem__. Work found in genreClassifier.ipynb under the heading “Creating a Custom Dataset Class for Spectrogram Dataset”.
- Poon, Matthew
    - Collaborated with all team members over Zoom in a paired-programming session to add the max_loudness feature to the model
    - Collaborated with all team members over Zoom as the typer (for half the session) in a paired-programming session to implement the recommender system. This is in Recommender.ipynb.
    - Created dictionaries for Track ID : Youtube URL and Youtube URL : Genre to be used in generating spectrograms (See code under heading “Creating a Dictionary of Songs and Youtube Links” in genreClassifier.ipynb)
        - Saved the above dictionaries using the Pickle library for later use
        - Collaborated with Aswin to merge the aforementioned dictionaries with his code to get spectrograms for improving our model and creating the recommender system
- Sai Subramanian, Aswin
    - Collaborated with all team members over Zoom in a paired-programming session to add the max_loudness feature to the model
    - Collaborated with all team members over Zoom as the typer (for half the session) in a paired-programming session to implement the recommender system. Work found in Recommender.ipynb.
    - Developed the spectrogram extraction process based on the Librosa tutorial at https://www.youtube.com/watch?v=3gzI4Z2OFgY ("How to Extract Spectrograms from Audio with Python"). This work can be found in genreClassifier.ipynb under the heading "Developing the Spectrogram Extraction Process".
    - Filtered the URLs in url_genre_dict for URLs most likely related to the music tracks we were looking for. This work can be found in genreClassifier.ipynb under the heading "Filtering Potential Audio Samples' YouTube URLs".

- Extracted and stored ~2000 spectrograms from 1063 URLs to form our new spectrogram dataset for genre classification. The code for this can be found in genreClassifier.ipynb under the heading "Generating and Storing Spectrograms".