Harry Verhoef, u1706021

A mobile application for democratic music playback with a neural network suggestion system.

Problem Statement

Generally, in places such as a gym, a party or a car, the music being played is controlled by one person or device, the “host”. This mobile application aims to replace this idea with that of a collaborative democratic music playback environment where each user has the ability to make their voice heard. Users will be able to vote for the next song, rate the current song and request to change the volume. While the song that is voted next is likely to be known by a large portion of the lobby, there will still be users who are unfamiliar with the democratically elected song. Therefore, the app will provide on-screen lyrics in time with playback. With these functionalities it is important to let the users in a lobby communicate, therefore each user will have the ability to enter a chat room. In an attempt to prevent users from abusing their voting powers, a vote-weighting system will be implemented so that users who have their recommendations downvoted by a certain percentage will have their vote worth less across all future lobbies. On the contrary, if their suggestions are perceived by other users to be particularly good, then their vote will become worth more.

In the event that no song has been decided as the next song, the obvious option would be to shuffle the playlist to find a pseudo-randomly generated song. The whole purpose of the app is to provide music playback with a high probability that the majority of users will enjoy. Therefore, the application will make use of a neural network to determine the next song in the queue if the vote results in a draw or there are insufficient votes. The inputs for this neural network will stem mainly from user’s most played tracks (if data is available) and the genres/songs most voted for in the past. Feedback will be provided to the network via the ratings of the songs suggested by the neural recommendation system.

The host device will be the one controlling the lobby settings, who is allowed to join the lobby, etc. Instead of having to manually enter a lobby key, host devices can send a deep-link to join the lobby or generate a QR Code.

Objectives

1. Host:
    - 1.1. The host user will be able to create a lobby and invite users to this lobby either through a key, a link, or a QR code.
    - 1.2. The host user will be able to alter the following lobby settings before and after creation:
        - 1.2.1. Lobby name
        - 1.2.2. Base playlist
        - 1.2.3. Suggestion mode (in-order, shuffle, neural)
        - 1.2.4. Maximum users
        - 1.2.5. Type of lobby (gym, party, car, etc.)
        - 1.2.6. Chat room (enabled/disabled)

        1.2.7. Volume Control (enabled/disabled)
        1.2.8. Lyrics (enabled/disabled)
    1.3. The host user will be able to terminate their lobby at any time.
    1.4. The host user will be able to override the decisions made by the other lobby members, and the whole lobby will be notified.
    1.5. The host user will be able to authenticate their account with Spotify accounts service.
2. User:
    2.1. Users will be able to join a lobby if they enter the correct key given it is not full.
    2.2. Users will not be able to join a lobby that is full.
    2.3. Users can only be in 1 lobby at a time.
    2.4. Users can leave a lobby at any time.
    2.5. Users can choose to show or hide the lobby chat room if enabled.
    2.6. Users can choose to show or hide the song lyrics if enabled.
    2.7. Users can thumbs-up or thumbs-down the current song
    2.8. Each user has 1 vote for the next song (however, their vote may be worth more or less than 1).
    2.9. A user's vote weight depends on the number of times they have received a significant number of thumbs-up or thumbs-down for a suggestion.
    2.10. A user cannot obtain a vote weight of 0.
    2.11. If the lobby settings allow it, users can vote to turn the volume up or down.
    2.12. The user can save any song being played to their Spotify library.
    2.13. Users from both Android and iOS platforms can download and use the app.
3. Miscellaneous:
    3.1. The backend API must run efficiently in order to be scalable, and therefore multithreading may need to be implemented.
    3.2. The app UI will be responsive in that it will function as expected on a wide range of mobile devices and tablets, regardless of screen size or aspect ratio.
    3.3. The lyrics displayed on screen will be in-sync with the song being played.
    3.4. Develop and use a test suite to determine the reliability and overall functionality of the application.
    3.5. The neural network will allow users to get smart recommendations on the next songs to play and will be called as a tie-breaker when two songs have an equal number of votes.
    3.6. The neural network will use the thumbs-up/thumbs-down data on songs it has suggested as part of its feedback, so that it learns as the app is being used.
    3.7. The neural network will be hosted on the external application server.
    3.8. Users will be able to download the app from the app store.

Methods

The mobile application will be developed through the use of the agile development methodology, where the software is incrementally developed and each release is planned for and then tested against said plan. Releases will be easily monitored using a private Github repository (private so that secret API keys are kept secret) and so that code cannot be copied by anyone. Once the project has been developed to a sufficient extent the repository will be made public so that it is open-source, and the secret API keys will be omitted. Requirements will be gathered throughout the development process using the objectives listed above and during the sprint processes, where a new requirement may become necessary or an old one redundant. With each sprint, a miniature test suite will be developed that tests the new release. This test suite will be developed with the sprint plan in mind. Towards the end of the development lifecycle a large test suite will be developed that will test each requirement with a wide variety of test cases.

I will be writing the front-end code in react-native (a JavaScript framework), however bridges between native modules will be needed in order to communicate with the SDKs on iOS and android. I will be programming these in Objective-C and Java, respectfully. For the backend the API will be written in node.js, another JavaScript framework. It is currently unsure as to what language will be used to write the neural network, this will be determined during the neural network research and design phases (task 13 and 14). A database will need to be utilised to store user data. For this, a mongoDB database will be used as a schema-less database to help increase scalability.

Timetable

Task No.	Task	Time taken	Date (Starting 05-Aug-19)	Dependencies
1	Setting up development environment.	1 week	12-Aug-19	
2	Creating temporary UI and researching react-native.	2 weeks	26-Aug-19	1
3	Integrating project with iOS Spotify SDK.	8 weeks	21-Oct-19	2
4	Designing and implementing a responsive and functional UI for beginning of user-story.	2 weeks	04-Nov-19	1
5	Implementing the ability for users to create and terminate a lobby.	1 week	11-Nov-19	4
6	Implement the ability to join/leave a lobby and vote for their preferred song.	1 week	18-Nov-19	2,4
7	Implement the switching of songs to the most voted.	0.5 weeks	18-Nov-19	6
8	Write up the progress report.	0.5 weeks	25-Nov-19	
9	Implement thumbs-up/thumbs-down on current song.	1 week	01-Dec-19	6
10	Implement a vote-weighting system.	1.5 weeks	11-Dec-19	9
11	Integrate with Genius API (for displayable lyrics)	2.5 weeks (Factoring in	28-Dec-19	4

		Christmas holidays)		
12	Implement the displayable lyrics during songs.	2 weeks	11-Jan-20	11
13	Researching neural networks.	1 week	18-Jan-20	
14	Designing the neural network.	1 week	25-Jan-20	13
15	Implementing the neural network recommendation system.	2.5 weeks	14-Feb-20	14
16	Implementing the chat room.	0.5 weeks	17-Feb-20	4
17	Creating and running the test suite.	1 week	24-Feb-20	
18	Preparing the oral presentation.	1 week	02-Mar-20	17
19	Integrate app with Spotify android SDK	1 week	09-Mar-20	2
20	Majority of final report.	6 weeks	20-Apr-20	17
21	Finishing touches on final report.	1 week	27-Apr-20	20

Resources

In order to develop and simulate and iOS app, physical iOS devices and a machine with mac OS are required, this will be needed from start to finish. In order to integrate the app with the Spotify android SDK, I will need an android device and Android Studio will need to be installed, in order to simulate android devices of different size and aspect ratio.

In order to learn about and design my neural network I will need access to “S. Rogers and M. Girolami, A first course in Machine Learning, CRC Press, 2011. Ch1”, which is available in the CS342 online material webpage.

I may also need to have an application server being run so that the project can access it’s own API constantly. This will be acquired through an online application server hosting service. This will be needed once the project will require testing of multiple users in a lobby while in different networks, and during the development of the neural network.

Risks

I only have access to one mac OS device and therefore if this one breaks I will not be able to continue iOS development until it is replaced, this would add potentially 2-3 weeks of project delay, depending on the development lifecycle with respect to the project timetable. As a backup strategy I will be able to set up a virtual mac OS emulator on my desktop PC from which I can plug in the iPhone and continue iOS development. If my laptop were to break then all my code would seemingly be gone, thankfully I will be using Github as version control and as a result I will be able to clone the repository to any new machine and continue from my most recent commit.

Another risk is that development is taking too long, in this case I would remove whatever I deem to be the least priority task out of the project timetable. For example, the lowest probability task is probably the chat room, this is additional functionality that, if not fully implemented, will not hinder the overall purpose of the application. This would then free up time that could be used completing a task that has higher priority.

Legal, professional or ethical considerations

I am aware that Spotify reserve the right to prevent the application from using both their SDK and web API. If this were to be the case then I would implement the project but with another music streaming service instead like amazon or youtube music (since both can be used on iOS and android). If I decided to commercialise the app then I would have to apply for a specific license from Spotify to do so. This is not the case and as such I do not have to worry about this application process.

With regard to the app being published on the app store, that is up to Apple, I will have to make my application completely compliant with their terms of service. Upon submission of an application, Apple normally take within 2 days to make their decision. I will be making these submissions of the app to Apple before the preparation for my oral presentation, so that I can potentially demo the app. In the unlikely case that Apple are persistent in rejecting my app, then I must be able to simulate users using the device so that during the demo my complete app can be demonstrated.