PACE THE MUSIC: ACTIVITY-BASED PLAYLIST GENERATOR

James Gray
University of Victoria
grayj@uvic.ca

Kaileen McCulloch
University of Victoria
kaileenm@uvic.ca

Nick Warwick
University of Victoria
nwarwick@uvic.ca

ABSTRACT

The following design specification outlines our plan to create a physical activity based playlist generation utility. It specifies the goals of our project, and provides a development timeline along with a list of tools required for development. This document also lists the roles of each team member, and provides links to any resources that we use during development.

1. INTRODUCTION

Many people enjoy listening to music while they exercise, and in many cases music itself can motivate the listener to put in even more effort than they would otherwise. [1] However, some types of exercise require strict adherence to a pre-existing training plan - for example, runners and cyclists often train at different paces for different portions of their activity. In these instances, the influence of music on an athletes pace or performance could be detrimental; a very fast song might subconsciously cause a runner to run at a faster pace than they intended to.

To address this problem, we plan to create a playlist generation utility with a focus on creating playlists for various types of physical activity such as running, cycling, or strength training. The user of this utility will input an activity plan, specifying features such as running pace or step frequency over the course of the activity. Given the plan as input, the playlist generation utility will create a playlist from an existing set of songs, basing the playlist on audio features such as tempo, genre, mood, and subjective energy level.

As an example, a user could create a running plan which specifies that they will run at a slow pace for fifteen minutes, increasing to a medium pace for twenty minutes, fast for ten minutes, and finally cooling down at a slow pace for five minutes. Given this input, the utility would generate a playlist containing songs at a slower tempo for the first fifteen minutes, moving to faster music as the pace of the run increases, following the user-created plan as closely as possible.

© James Gray, Kaileen McCulloch, Nick Warwick.
Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** James Gray, Kaileen McCulloch, Nick Warwick. "Pace the Music: Activity-Based Playlist Generator".

2. RELATED WORK

Playlist generation is a popular topic in the field of MIR. Creating the right playlist to supplement an activity is a complex problem, and automating the process is even more so. There are a wide variety of external factors that can be used for playlist generation. For example, Oliver and Kreger-Stickles created a generator which bases the decision process on the users physiological response [2]. Similarly, Pauws and Eggen studied the correlation between the environment and the type of music a given user would listen to in that environment [3]. Environment-based playlists are also present in streaming music applications such as Google Play Music and Spotify, which allow users to choose playlists based on their location - for example, in the shower or at a coffee shop. Additional factors and methods in playlist generation include random shuffle, content [4], frequency spectrum, and skipping behaviour [5].

Our idea of generating playlists based on user-specified activity plans is similar to the work done by Masahiro et al. [6] and Chen et al. [7]. Masahiro et al. worked on the development of an automatic music selection system based on a runners step frequency, and Chen et al. developed a music assisted run trainer which also uses step frequency to slow down or speed up the tempo of the music to match the user's pace. We plan on using step frequency as one of the decision features in our algorithm; however, the step frequency we plan on using will not be a real-time measurement, but rather a calculation based on the user-specified activity plan.

Tempo tracking and audio feature analysis are other key components of our research. Musical tempo has been identified as an important component of music classification [8] [9] [10]. Other audio feature extraction techniques which have been explored are intensity, timbre, rhythm, and mood tracking [11] which may also play a role in our playlist generator's selection algorithm.

3. TIMELINE

The following development timeline outlines the major stages of the project and their estimated completion dates.

1. **Determine goals:** [Complete] During this phase we determined the goals for the end product and what capabilities it should have. We also decided on a platform for the application. (*February 22nd to 23rd*)

2. **Requirements gathering:** [Complete] We developed functional and non-functional requirements for the

project. These outline our functionality goals for the final product. (*February 23rd to 26th*)

3. **Design:** [Complete] We decided on a system layout. After a system layout was determined we began development on the UI and started structuring the backend of the application. (*February 26th to March 1st*)

4. **Prototyping:** [In-progress] We are in the progress of completing a rough version of the application with basic functionality. (*March 1st to 21st*)

5. **Testing and revision:** [In-progress] Throughout development we have repeatedly tested the prototype to ensure that it is working as intended and that each requirement has been met. Once a final draft has been created, a thorough testing set well be run to ensure no final issues need to be addressed. (*March 17th to April 1st*)

6. **Release:** [To do] Complete development of the final version of the application. (*April 1st to 5th*)

4. TOOLS AND RESOURCES

The project will require a range of tools and technologies, from audio feature extraction and analysis tools to database utilities, application and UI logic.

4.1 Backend

We intend to program the core application logic with Python. This code will interface between the various layers of the utility, such as the core playlist generation logic, the database, the audio feature extraction logic, and the UI code.

4.2 Database

To create a playlist, the utility will need access to a set of songs. The song files will be stored on disk, and any additional metadata or audio features will be stored alongside the files in a database. We intend to use a MySQL database alongside a Python database interaction utility such as SQLAlchemy [12]. Additionally we intend to build a corpus of song files using Music21 [13].

4.3 Audio Feature Extraction

The audio files themselves will need to be analyzed in order to extract the necessary features from the music, such as BPM information. We plan to use Marsyas and the associated Python bindings to extract this information, which will then be stored in the database in order to refer to it later.

4.4 GUI

Our current plan is to create a standalone desktop utility or application. The GUI for this application will be created using a Python UI library or framework, such as PyJamas [14], PyQT [15], or TkInter [16].

5. PROGRESS REPORT

5.1 Completed Work

As of March 19, we have made significant progress in developing the foundation of our playlist generation utility. Most of the base data model has been completed, with core object classes implemented, and the associated table schemas and relationships have been defined using SQLAlchemy. These classes range from music-related classes, such as Artist, Song, and SongMeta, to activity-related classes, with definitions for ActivityPlan, Pace, and Segment objects. Additionally, basic database setup logic is in place to both create and drop the database and its tables, allowing the utility to be installed and run quickly after downloading. The base layout for the UI has been completed, in terms of where the buttons and widgets will be placed, with some basic functionality in place as well. Some exploration into audio feature extraction using EchoNest has been done, with a proof of concept established using API calls to query the service for tempo and energy information for different songs; however, we decided to eschew the service in favour of using Marsyas to perform feature extraction locally, in order to avoid expensive API calls. Core app configuration logic has been implemented, as well as some basic documentation describing the dependencies and installation instructions.

5.2 Remaining Work

With the base layout of the UI complete, we can start working on adding functionality to the UI. This will involve taking user input to create objects which represent "segments" of a given playlist. These objects will be stored in a list, which will be repeatedly updated as segments are added or removed. This list will then be displayed to the user in the UI. Once functionality for the UI has been finished, we can begin to tie the different parts of the application together by adding interaction between the front-end and back-end of the application. This will involve persisting user actions to the database; for example, if a user removes a segment from a playlist by performing an action on the UI, that segment should, at some point, be removed from the corresponding playlist in the database by committing the changes. Once the functionality has been tied together, we will have a working prototype available for testing. The testing phase will likely uncover bugs that will require minor changes to be implemented. These changes could involve modifying the layout of the UI, as the base version has not been tested with any functionality. Although the back-end of the application has been lightly tested, more extensive testing will be required once the different areas of the application have been tied together.

5.3 Revisions

Once we started working on the project, it quickly became clear what aspects of the original proposal would be suitable and which aspects would need to be changed from the original plan. We specified in the project proposal that we would generate a music dataset using Music21; instead, we

decided for simplicity's sake to use a dataset composed of a variety of music from our own libraries. Once we had our dataset, we needed to determine exactly what audio features to extract in order to match the songs up with exercise pace. In order to do this, we tried using EchoNest to classify the music. EchoNest provides a list of many different audio features, such as energy, danceability, and BPM. After trying out EchoNest, we decided to use only BPM and to extract it using Marsyas, for reasons outlined previously. In terms of front-end development, the UI framework is being implemented using PyQT due to its cross-platform support. In terms of back-end functionality we chose to use SQLite instead of MySQL because it is simpler and does not require the user to install a large database system. MySQL is a good choice on a server-based web application; however, it is not intended for use on applications that run directly on a clients machine. SQLite is much more lightweight and serves our purposes better with regards to keeping the application small. We initially started our back-end implementation using Python 3; however, the front-end development had been started using Python 2.7, so we collectively decided to switch all development to Python 2.7 due to our familiarity with that version of the language. Despite these revisions, we have still met all of our timeline goals, with the exception of the prototype deadline. However, due to the release schedule being extended to April 13th, we intentionally pushed the remaining deadlines back a week to allow ourselves more time to perfect the prototype.

6. TEAM ROLES

The roles of each team member are loosely defined as follows; however, they are subject to change, and team members will contribute in other areas of the project as needed. Nick will be in charge of GUI design and front-end development, James will handle back-end design and database logic, and Kaileen will be in charge of developing feature extraction logic.

7. CONCLUSION

While plenty of existing work has addressed the problem of playlist generation, activity-based playlist generation remains a relatively unexplored problem. Using the tools and resources outlined above, we intend to create a powerful, user-friendly utility that will provide some helpful insights into the topic of activity-based playlist generation. We feel condifent that our current progress, combined with the work we will accomplish in the coming weeks, will result in an effrective system that fans of both music and fitness will love.

8. REFERENCES

[1] N. Shivar. "How I Cut 1:57 Off My Average 5k Time By Tweaking My Playlist." http://www.nateshivar.com/1182/how-i-cut-157-off-my-average-5k-time-by-tweaking-my-playlist/, 2015.

[2] N. Oliver and L. Kreger-Stickles. "PAPA: Physiology and Purpose-Aware Automatic Playlist Generation." In *Proceedings of the International Symposium on Music Information Retrieval.* http://ismir2006.ismir.net/PAPERS/ISMIR06162_Paper.pdf, 2006.

[3] S. Pauws and B. Eggen. "PATS: Realization and User Evaluation of an Automatic Playlist Generator." In *Proceedings of the International Symposium on Music Information Retrieval*. http://www.ismir2002.ismir.net/proceedings/02FP074.pdf, 2002.

[4] B. Logan. "Content-Based Playlist Generation: Exploratory Experiments." In *Proceedings of the International Symposium on Music Information Retrieval.* http://www.ismir2002.ismir.net/proceedings/03SP052.pdf, 2002.

[5] E. Pampalk, T. Pohle and G. Widmer. "Dynamic Playlist Generation Based on Skipping Behavior." In *Proceedings of the International Symposium on Music Information Retrieval.* http://cis.ofai.at/~elias.pampalk/publications/pam_ismir05b.pdf, 2005.

[6] N. Masahiro, H. Takaesu, H. Demachi, M. Oono and H. Saito. "Development of an Automatic Music Selection System Based on Runners Step Frequency." In *Proceedings of the International Symposium on Music Information Retrieval.* 2008.

[7] L. Chen, Y. Tung and J. R. Jang. "MART: Music Assisted Run Trainer." In *Proceedings of the International Symposium on Music Information Retrieval.* http://www.terasoft.com.tw/conf/ismir2014/LBD/LBD24.pdf, 2014.

[8] M. McKinney and J. Breebaart. "Features for Audio and Music Classification." In *Proceedings of the International Symposium on Music Information Retrieval.* https://jscholarship.library.jhu.edu/handle/1774.2/22, 2003.

[9] A. Pikrakis, I. Antonopoulos and S. Theodoridis. "Music Meter and Tempo Tracking from Raw Polyphonic Audio." In *Proceedings of the International Symposium on Music Information Retrieval.* http://www.ee.columbia.edu/~dpwe/ismir2004/CRFILES/paper160.pdf, 2004.

[10] M. Alonso, B. David and G. Richard. "Tempo and Beat Estimation of Musical Signals." In *Proceedings of the International Symposium on Music Information Retrieval.* http://www.ee.columbia.edu/~dpwe/ismir2004/CRFILES/paper191.pdf, 2004.

[11] D. Liu, L. Lu and H. Zhang. "Automatic Mood Detection from Acoustic Music Data." In *Proceedings of the International Symposium on Music Information Retrieval.*

https://jscholarship.library.jhu.edu/handle/1774.2/14, 2003.

[12] “SQLAlchemy.” http://www.sqlalchemy.org, 2016.

[13] “Music21.corpus.” *Music21 Module Reference.* http://web.mit.edu/music21/doc/moduleReference/moduleCorpus.html, 2015.

[14] “Pyjs.” http://pyjs.org/Overview.html, 2016.

[15] “PyQT.” https://www.riverbankcomputing.com/software/pyqt, 2015.

[16] “TkInter.” http://tkinter.unpythonic.net, 2014.