Project Report: Dynamic Playlist Generation (December 2020)

Gus Campbell, Ananya Kansal, Jocelyn Kavanagh, and Rose Sun

Abstract—**In this paper, we describe a project that will determine an effective method for generating a music playlist based on contextual data. The motivating use case for this project is a restaurant owner's desire for personalized, adaptive playlists to enhance their restaurant atmosphere. By applying past studies on the connection between music and food consumption, the proposed playlist generation system hopes to enhance consumer satisfaction, while maintaining autonomy from the restaurant owner, allowing them to focus on other tasks. We created this system by developing algorithms in Python that map contextual data, like user preference and sound power level to song metadata. The project's conclusion is marked by the completion of a functional prototype in form of a web application.**

I. Introduction

IN the modern world of digital music streaming, context-based listening is gaining popularity on streaming services. Services such as Spotify, Google Play, Apple Music, and many others have implemented music recommendation systems that generate playlists optimized for different situations and moods. However, the auditory experience is often ignored in everyday settings such as restaurants. Music plays in the background of many restaurants, but the song selections are usually repetitive and are not well suited to the environment. In this paper, we propose an idea to incorporate some of the technology used to create generative playlists with active monitoring technology to generate "smart" playlists that would enhance a restaurant outing.

II. Motivation

The auditory experience of a consumer is vital to their satisfaction. In the restaurant industry, studies have shown that music impacts the way that people consume and taste food and the way they spend money [1]. Tempo has shown an effect on eating patterns with high-volume, high-tempo music causing customers to eat more rapidly, leading to faster table turnover [4]. Frequency effects how food tastes with higher tones enhancing sweetness and lower tones promoting bitterness [2]. Musical genre and style may give the impression of authenticity and expense [1][3], and studies show that playing random chart music as opposed to curated songs can decrease sales by 9.1% [5].

Despite the relationships shown between music and food consumption, the restaurant industry has not fully realized music as a resource to boost consumer satisfaction and sales. Typically, restaurants pay for licensing services and play top hits, unspecific playlists, or radio stations. Even though music's importance has started to come to focus on the restaurant industry, via anecdotal evidence, music is not being used efficiently or to its full capacity. Our team believes that this is due to lack of time, resources, and data. Though restaurant owners may have a specific musical goal in mind, they do not have time to curate a personal playlist, and they do not have time to constantly monitor and adjust a playlist during operating hours. Our team developed a prototype for a product that would alleviate a restaurant owner's burden of playlist creation while maximizing the benefits of personalized music in a restaurant setting.

A. *Related Works*

To explore the topic of context aware recommender system and playlist generation, we studied survey articles [7] and [8]. Traditional recommender systems can be significantly improved when incorporated with context-aware recommender. Currently, there are few products on the market for playlist generation for businesses, and those available are relatively static. To maximize consumer experience while maintaining autonomy for the owner, our team moved to enhance the current product ideology by adding a dynamic component. Specifically, we added an auditory monitoring system that assess the noise of the restaurant and uses that data to actively select future songs.

1) *Soundtrack Your Brand*

Soundtrack Your Brand [6] provides businesses personalized "soundtracks" based on background user data. Users can choose from "expertly curated" playlists, play through artist radio stations, or they can select song tags such as energy level and mood to generate a personalized playlist. These playlists can be adjusted given user feedback and are played via a phone application. Soundtrack Your Brand also provides the licensing required to stream music in a business.

III. Goals

Our project aimed to create a "smart" playback system that plays situationally appropriate music given a user profile and auditory monitoring. The overarching goal was to accurately map contextual data to song metadata in order to create a customized musical experience. The system intends to function autonomously with real-time capabilities to allow for user interaction. In practice, we want a final product that creates better experiences for both business owners and their consumers.

IV. Method and Decisions

Our prototype functions as a web application launched at the command line. The backend of the application is written in Python utilizing the Flask framework to run on a web server. Given the Flask framework and the need to service between Python and HTML, the software design attempts to follow the model view controller pattern.

The application utilizes the MediaEval AcousticBrainz Genre dataset [9]. This dataset was captured from the Discogs website, an editorial database maintained by music experts and consumers. From the dataset, we extracted 44530 songs and found their respective metadata with the AcousticBrainz API.

We classified the songs into 172 static playlists based on different combinations of parameters in the retrieved metadata (including genre, mood, timbre, instrumentation, tempo and danceability) and stored them in csv format. Based on user inputs collected from the GUI, the playlist that matches the user's parameters is located and can be played in three different ways: shuffle, similarity-based, and room noise-based. In all cases, as one song is played the next is loaded into the queue. There are only 2 songs in the queue at a time, and for both the similarity based and room-noise based methods, on the previous song is utilized in the selection algorithm. With the use of dequeue and list data structures, no song is played more than once per playlist generation.

A. Updated Goals and Timeline

We initially designed 3 stages for our project: playing through a static list, playing through a list with adjustments according to sound pressure level, and playing through a list with adjustments according to sound pressure level and real-time user input. We anticipated that the initial user data needed for playlist generation includes type of restaurant, energy level of the restaurant, average time a customer spends in the restaurant, and preferred genres. We anticipated sound pressure level to influence the volume of playback and the valence/intensity of the song selection, and the real-time user feedback to specify why the user didn't respond well to selected songs to refine future song selection. We decided to use clustering method or weighted similarity matrix for playlist generation.

As we faced challenges, our goals and stages changed. Most notably, we dropped the final stage of our initial proposal, adding real-time user input. We determined that under the time constraints, we decided to re-design the GUI so that the user could enter and re-enter preferences to change their playlists selection anytime. This replaced the final stage real-time user input, allowing us to focus on the first two stages before and after the midterm presentation. Below is our updated timeline for this project.

1) *Week 1-6 – September 14 - October 19*
   - Extracted metadata from the dataset
   - Developed playback engine with play/pause functionality and simple soundtrack information display
   - Implemented GUI for playback engine
   - Experimented with different playlist generation methods
2) *Week 6-11 – October 19 – November 23*
   - Classified and stored static playlists
   - Developed playback engine with advanced soundtrack information display
   - Implemented GUI for user feedback during playback
   - Wrote the function for detecting the sound power level
   - Implement the algorithm that maps sound power level to tempo within static playlists
3) *Week 12-14 – November 30 – December 7*
   - Tested and improved the algorithm that maps sound power level
   - Refined and enhanced areas of the GUI
   - Connected individual parts
   - Final testing and adjustments

B. Summary of Changed Features

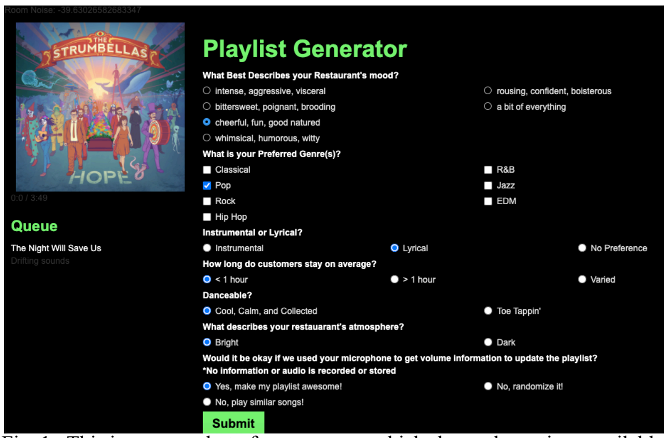
As mentioned above, we used the MediaEval dataset along with the AcousticBrainz API instead of building our own corpus. We determined that developing our own corpus was not worthwhile due to the lack of diverse and available songs in our personal collections. The MediaEval dataset was captured from the Discogs website, an editorial database maintained by music experts and consumers. We chose to use this dataset because it was generated from a consumer list, and it already matched the songs to their respective MusicBrainz ID. Additionally, because we did not play from a local library, we needed another playback method. We moved to use the Spotify Web API and Spotify Web SDK. We retrieved the unique Spotify URI's for each song in the dataset via the web API, and we play through them using the SDK.

Instead of using a service like Tkinter for our GUI, we decided to use Flask to create a web application. This means that our final prototype's GUI is a webpage written in HTML and stylized with CSS.

We implemented our own function for sound pressure level using PyAudio instead of using an external command-line tool. The average sound pressure level is mapped to the tempi of the selected songs. When the sound pressure level increases, the queued songs are faster, and when the sound power level decreases, the queued songs are slower. We chose this mapping based on our preliminary research and our own intuition.

All other metadata is explicitly mapped to the user's selection. We initially hoped to avoid direct mappings from the user's preferences (instead of directly asking instrumental/vocal, ask a restaurant related question), but we determined this made the GUI too vague and hard to use.

As mentioned above, we added another user option for the method of song selection. The user has the option to play via shuffle, with a similarity algorithm, or with the average sound pressure level. This allowed us to use all approaches of playlist generation that we explored, and it gives the user more options while interacting with the application. The shuffle mode randomly pulls songs from the playlists and appends them to the queue. The similarity-based mode randomly pulls the first song to be played, calculates a similarity matrix using the metadata and pairwise cosine similarity, and appends the song that is the most similar to the previously played song to the queue. The room noise-based method maps the sound pressure level of the surrounding to the tempo of the songs. When the sound pressure level increases, faster songs get appended to the queue, and when the sound pressure level decreases, slower songs get appended to the queue. Consistent sound pressure level input would not necessarily result in the pick of one song repeatedly.


Room Noise: -39.63026582683347

0:0 / 3:49

Queue

The Night Will Save Us

Drifting sounds

Playlist Generator

What Best Describes your Restaurant's mood?

- intense, aggressive, visceral
- bittersweet, poignant, brooding
- [✓ cheerful, fun, good natured]
- whimsical, humorous, witty
- rousing, confident, boisterous
- a bit of everything

What is your Preferred Genre(s)?

- Classical
- [✓ Pop]
- Rock
- Hip Hop
- R&B
- Jazz
- EDM

Instrumental or Lyrical?

Instrumental [✓ Lyrical] No Preference

How long do customers stay on average?

[✓ < 1 hour] > 1 hour Varied

Danceable?

[✓ Cool, Calm, and Collected] Toe Tappin'

What describes your restauarant's atmosphere?

[✓ Bright] Dark

Would it be okay if we used your microphone to get volume information to update the playlist?
*No information or audio is recorded or stored

- [✓ Yes, make my playlist awesome!]
- No, randomize it!
- No, play similar songs!

Fig. 1. This is a screenshot of our program which shows the options available for user selection. The first question is mapped to the mood metadata, the second question is mapped to the genre metadata, the third question is mapped to the instrumentation metadata, the fourth question is mapped to the tempo metadata, the fifth question is mapped to the danceability metadata, and the sixth question is mapped to the timbre metadata. The final question determines the method for song selection.

C. *Challenges Faced*

In classifying the static playlists, we had to avoid very short or non-existent playlists generated due to heavily detailed parameters by merging some of the short playlists together. In creating the similarity-based mode algorithm, we had a hard time deciding which pairwise similarity method to use. With subjective listening tests, we decided to use cosine similarity method for that it yields the most similar result.

We were also forced to narrow our mapped feature down to tempo due to the complicated nature of multiple mappings. One problem we encountered was due to the rigid direct mapping--when the sound power level input stayed the same, the algorithm would always pick one song within a static list repeatedly. Our solution was to add a Gaussian white noise to the sound power level value to be mapped for it to vary within a desired small range, and we applied a random pick within this small range.

We also expect some complications with capturing the sound pressure level due to lower-quality on-board microphones and the sound source location of the music playback. We do not have enough on-site testing to claim this feature is completely effective, but we believe we have programmed it to succeed with any built-in equipment.

We faced challenges using Python Flask while asynchronously updating the GUI. The Flask framework is traditionally used with web applications that allow for website reloading. Because of the integration with the Spotify SDK, our web application had to serve on one, single-loaded page to avoid breaking the connection with Spotify. As such, we had to explore jQuery and AJAX to asynchronously update specific elements of the HTML. This proves to work well, however, it violates rules of the model view controller design pattern. Additionally, because we are running Flask on a development server, there are a few unexpected bugs related to the server unexpectedly halting.

D. *Roles Explained*

Software Development

Responsible for developing algorithms/mapping paths, querying the song database, and creating the playback engine. Included researching tools and libraries utilized during the project. In the case of this project, every member on the team assisted in this role at some point in the timeline. Co-led by J. Kavanagh and R.Sun.

Front-End Development

Oversaw the implementation and integration of the GUI. This team is responsible for how users interact with the system and how the product looks when published. Led by A. Kansal.

Music Research

Handled the qualification of the music and works closely with the software development in the creation of mapping paths. This team determines which parameters should be specified and what features should be extracted by the software. Led by R. Sun.

Public Outreach

Handled all external communication, including the interviewing of restaurant owners and communication with Dr. Lerch. Led by G. Campbell.

QA & Testing

Responsible for testing all facets of the project, as well as ensuring system outcomes match test users' expectations. Led by G. Campbell.

V. Evaluation

Based on our project goals originally set in our proposal, we believe that this project was a success. Originally, we stated that our project should generate playlists that align with and adapt to the user and environmental data gathered. Each algorithm should yield appropriate results given our contextual data mappings in a timely and efficient manner. And despite a lack of independent testing, we, as a group, believe these criteria to be met to our expectations. As a result (given the available resources and amount of time allotted), we now base success on whether the algorithm is working as intended. The interface looks/works above expectation, including input fields for user preferences and displayed song information. Through internal testing, we find that the success criteria are satisfied, and the program fulfills the goals set in the original proposal.

VI. Future Opportunities

Despite an overall success, we would like to make improvements to this project in the future. Specifically, we would like to explore additional feature mappings with the sound pressure level input, and we would like to test the system with real users. This would help us inform appropriate mapping decisions and make a more accessible GUI. We would like to implement stage 3 (as outlined in the initial proposal) by adding a dynamic like/dislike feature to the GUI to improve user experience. Finally, despite testing, there are still bugs in the software. We hope to resolve some of the issues by moving away from a development server, as well as through more rigorous testing. We will also put continuous work into improving the efficiency and security of this program.

VII. Resources

A. *Tools and Languages*

The primary language used in this project is Python. Tools and frameworks developed for use within the Python environment including native packages such as json and threading, and external packages such as requests, Pandas, Scikit-learn, Flask and PyAudio were utilized.

B. *Libraries and APIs*

We used the MediaEval AcousticBrainz Genre dataset [9]. This dataset was captured from the Discogs website, an editorial database maintained by music experts and consumers. From the dataset, we extracted 44530 songs and found their respective metadata with the AcousticBrainz API. The songs are played via the Spotify Web API and Spotify Player SDK.

References

[1] C. Caldwell *et al.,* "Play That One Again: the Effect of Music Tempo on Consumer Behaviour in a Restaurant," *E-European Advances in Consumer Research*, vol. 4, pp. 58-62, 1999.

[2] Q. Wang, "Music, Mind, and Mouth: Exploring the Interaction Between Music and Flavor Perception.", unpublished. *Available: https://www.media.mit.edu/publications/music-mind-and-mouth-exploring-the-interaction-between-music-and-flavor-perception/*

[3] G. Pezzzini., "5 Reasons Why You Should Play Music in Your Restaurant," *LS Retail*, Dec. 2017, [Online]. *Available*: https://www.lsretail.com/blog/play-music-your-restaurant

[4] "How Music Establishes Mood and Drives Restaurant Profits," *National Restaurant Association*, Jul. 2015, [Online]. *Available*: https://restaurant.org/articles/operations/how-music-establishes-mood-and-drives-restaurant-p

[5] K. Vandette., "Restaurant Music Actually Affects What You Order, Study Finds," *Earth.com*, Dec. 2017, [Online]. *Available*: https://www.lsretail.com/blog/play-music-your-restaurant

[6] "Music in Restaurants: From Beginner to Pro in Less Than 5 Minutes," *Soundtrack Your Brand*, [Online]. *Available*: https://www.soundtrackyourbrand.com/guides/music-in-restaurants-guide

[7] I. A. P. Santana and M. A. Domingues, "A Systematic Review on Context-Aware Recommender Systems using Deep Learning and Embeddings," *arXiv:2007.04782 [cs]*, Jul. 2020, Accessed: Dec. 06, 2020. [Online]. Available: http://arxiv.org/abs/2007.04782.

[8] G. Bonnin and D. Jannach, "Automated Generation of Music Playlists: Survey and Experiments," ACM Computing Surveys, vol. 47, pp. 1–35, Jan. 2014, doi: 10.1145/2652481.

[9] MediaEval AcousticBrainz Genre Dataset [Online]. *Available:* https://zenodo.org/record/2553414#.X81bEy2ZNQI