Join my party! How can we enhance social interactions in music streaming?

Alo Allik, Florian Thalmann, Cornelia Metzig, Mark Sandler
Centre for Digital Music, Queen Mary University of London
{a.allik, f.thalmann, c.metzig, mark.sandler}@qmul.ac.uk

ABSTRACT

In this paper we examine ways to encourage social interactions in online music streaming platforms and discuss the challenges that emerge when deploying browser-based music mixing systems. With the example of an interactive online application, that allows users to choose music collaboratively based on mood, create their own personal parties, as well as share their favourite tracks with other participants, we explore new alternatives for musical experiences in the context of social media on the one hand and music streaming on the other.

1. INTRODUCTION

With the streaming paradigm becoming the prevalent method of our daily musical experiences there are potential new ways to introduce more collaborative and socially interactive experiences to music listening in the age dominated by various social media platforms. Collaborative playlists have become standard on most streaming platforms, yet most music sharing seems to happen through posted links in social chat applications. We explore a way to select and share music collaboratively in a sample web application using higher level musical concepts such as mood. The selected music is automatically DJ mixed using the Web Audio API with the help of content-based audio features that assist in matching tempo, key, beats and volume of the tracks to create a continuous and evolving stream.

The sample web application we are going to use as a demonstrator of the different ideas explored in this paper, **moodplay.github.io**, has grown out of two earlier, now defunct music players. The original system - Moodplay[3] - was designed as an interactive installation for public spaces where participants could experience and interact with the system and each other on location. This made for an immediate and engaging experience, but at the same time significantly constrained public accessibility for the same reasons. It also involved remarkable time and effort to deploy, involving audio and visual systems, and complex interaction between components that made it unfeasible for any sort of frequent deployment. This Moodplay should not be confused with its namesake web application accessible at http://moodplay.pythonanywhere.com/, which is not related to the family of music players described here, yet contributes an interesting and valuable approach to the area of music mood similarity[2]. A web application also happened to be the next iteration of our Moodplay[1] in attempts to solve the problem of public accessibility, while also focusing on personalisation, discovery and playlist creation. However, various issues with music licensing, deployment infrastructure and, most importantly, the loss of the social aspect of the application forced the development to be abandoned.

Having been faced with the joys and challenges of developing and deploying these applications, a larger picture has gradually started to intrigue about how we listen to and share music in an era of music streaming services and social media, in which we are perpetually connected online and what kind of opportunities could this environment potentially afford. How users on various streaming services and online stores select music depends on the interface they are presented and affects their choices. We are interested in exploring if there are viable alternatives to the established ways of designing music selection interfaces and how to combine music selection with social aspects of listening to music, in particular, how to make music selection a collective interaction and how to enrich the sharing experience in web environments.

2. MUSIC SELECTION

The most common music selection interfaces on digital platforms, be it offline music players, online streaming services or retail stores, present choices on the level of artists or tracks in form of lists. Even curated playlists that usually include tracks on higher level notions of popularity, style, genre, mood or activity use the same format. A number of alternative representations of musical tracks or artists could help break from the standard by displaying a collection of such entities in the shape of networks, clusters or scatter plots, for example. The organising principle behind these kinds of displays would derive from some kind of similarity measure based on collection and analysis of contextual or content-based information. Figure 1 illustrates a network layout of a hypothetical track collection, in which each node represents a track, the colors of nodes identify an artist, the size of nodes could indicate popularity of each track, the connections are made based on content-based similarity, which could also serve as a probability weight when selecting a next track, for example. Large collections of data about music tracks, from which similarity models could be designed, can

Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.

Figure 1: Tracks could be displayed as a network in a music player interface.

be accessed from various publicly available sources. There are a number of music-related APIs, both community-run and commercial, from where rich troves of information can be gathered about musical entities like tracks, albums and artists, that can then be used to enrich the music discovery and selection for users. Connecting various data sources can also have a qualitative effect on musical experiences. Discogs[1], Gracenote[2], MusicBrainz[3], AcousticBrainz[4], MuSicStory[5], and Spotify[6] are just a few specifically music-related data sources that a music player could rely on for information, and there are many more.

In case of moodplay.github.io, the arrangement of the tracks, shown in Figure 2, is inherited from the original system and is based on crowd-sourced data from the ever popular Last.fm API (https://www.last.fm/api). Despite the gradual reduction of the site's functionality over the last few years, it still provides valuable information about how millions of users have described these entities by providing various tags related to genre, mood, instrumentation, geographical location, lifestyle and many other typical notions, but at the same time almost anything expressing a personal opinion. The tracks are arranged in the space by two mood coordinates: horizontally from negative to positive (designated as valence in dimensional models of emotion) and vertically from calm to excited (also arousal or intensity). The process that converts user tagging data to the scattering of tracks in the space involved first determining a set of most often occurring mood tags, which can also be extracted from the service. Then querying the API for counts of how many times each tag has been applied by users to each track produced a multidimensional space, in

[1]https://www.discogs.com/developers/
[2]https://developer.gracenote.com/web-api
[3]https://musicbrainz.org
[4]http://acousticbrainz.org/data
[5]http://developers.music-story.com/developers
[6]https://developer.spotify.com/documentation/web-api/

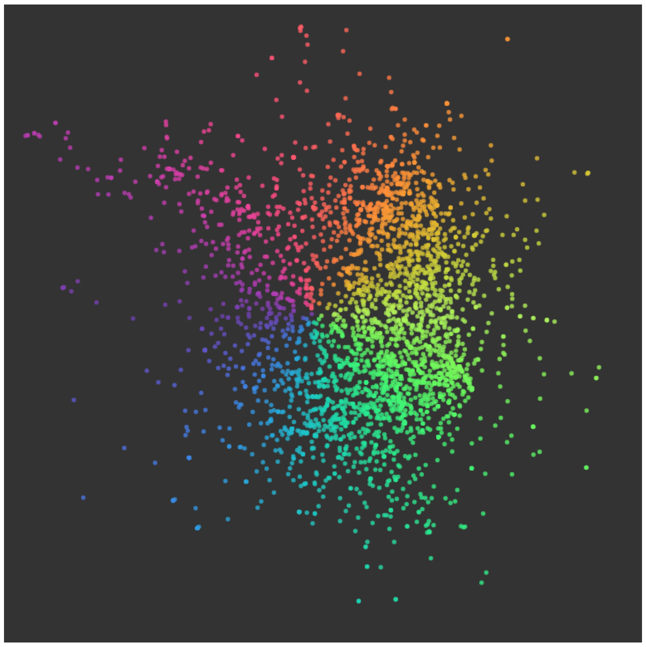

Figure 2: Representation of moodplay.github.io tracks in 2D space, arranged in terms of general mood from negative (left) to positive (right) horizontally and calm (bottom) to excited (top) vertically.

which each track is associated with a vector of tag counts. This enables calculating coordinates for each track in a 2D space by applying dimensionality reduction techniques. The full process of track selection together with the models of emotion and valence-arousal mapping of tracks has already been discussed more in depth in previous Moodplay-related papers.

The audio content for the player originates from Deezer API[7]. Due to licensing limitations, the current system can only access the 30-second previews of each track. The dataset included in the current version of Moodplay is a subset from the original system, that has been whittled down through a process of finding matching Deezer identifiers of tracks and further by including only the ones that have a valid preview URL. In the end there are 3,497 tracks by 2,314 artists left of the original collection.

Users can explore the mood space by selecting a location on the interface and are then displayed the corresponding mood tag for that area. The space is tessellated using the Voronoi algorithm[8] with the mood tag coordinates as the set of points as shown in Figure 3. The tessellation does not appear in the interface (which is described in more detail in Section 3 and shown in Figure 4), but partitions the space into mood regions in the background. Once a desired mood has been located, it can then be selected by clicking on the popup label, which means the user preference is communicated to the server and added to other users'. Thus users never select music directly by artist or track, but by emotion that the closest track to the preference matches.

[7]https://developers.deezer.com/api
[8]https://en.wikipedia.org/wiki/Voronoi_diagram

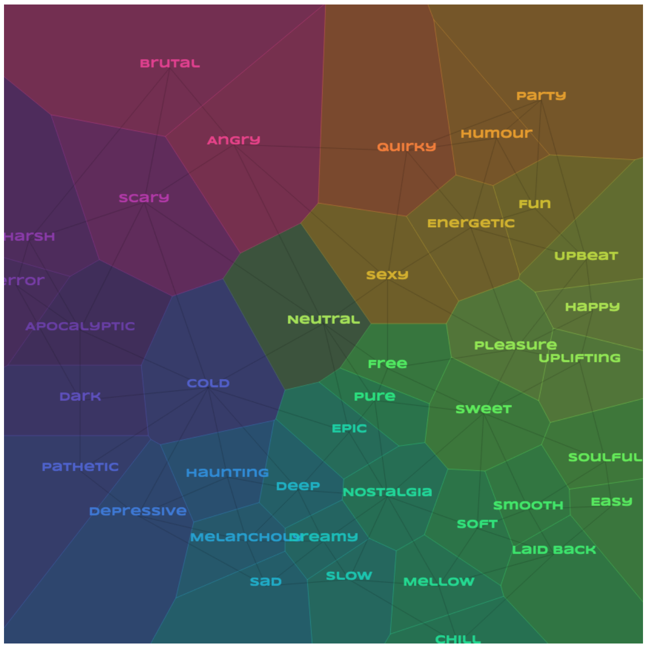

Figure 3: Voronoi tessellation of the 2D mood space that underlies the mood selection interface

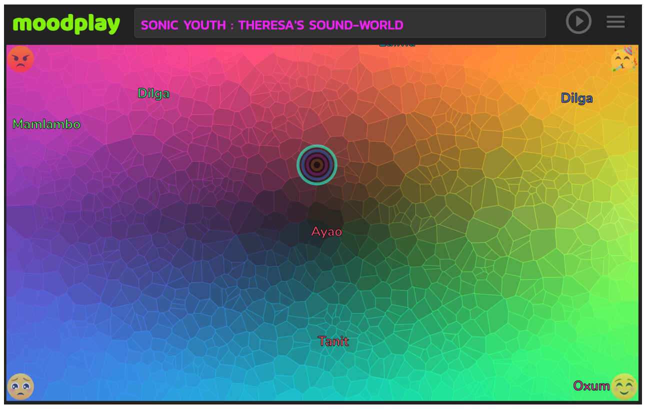

Figure 4: moodplay.github.io interface with 5 active user names displayed alongside the circular player cursor indicating the average mood of the party

3. SOCIAL INTERACTION

Social Web platforms accessible to anyone with a portable device have drastically changed the way we communicate and interact with each other. This also provides new opportunities in the context of music streaming, that perhaps have not yet been explored to their full potential. Collaborative music selection and sharing could be enhanced and encouraged through new types of music player interfaces that we discussed in the previous section. Social interaction on popular music streaming platforms is currently for the most part encouraged through collaborative playlists. Spotify used to provide a messaging functionality in the early versions that allowed users to send music links to each other, but that functionality was removed. Dubtrack.fm[9] is a social online radio where users can create their own rooms or join existing ones and collaboratively queue tracks they would like to listen to. Each rooms has a specific theme that represents a community of music listeners, like ”chillout” or ”The Eighties” for example, and code of conduct including what to play, what not to play, and active hours. Tracks can be queued from public music services like YouTube and SoundCloud. Playlist[10], a social music platform for iOS only available in select territories (i.e. not UK), enables, in addition to collaborative playlist creation, a synchronised listening and a chat functionality to participants. These are fine examples of communal music sharing that enhance user participation and music discovery, although still centred around the notion of a linear playlist of tracks as the core musical entities. An alternative to creating cumulative linear playlists could be based on consensus and involve users voting for their preference from a set of options. If we envisioned a network of tracks as suggested in the previous section, the users could be presented a selection of tracks to choose from based on feature similarity and the music stream is determined by which tracks receive the most votes. If the selection happens by a higher-level concept such as mood, genre or activity, for example, then the selection of tracks is determined by the system that aggregates user preferences, like in moodplay.github.io. Since the Moodplay web application can be accessed by any number of users simultaneously, the interaction becomes more immediate between potentially larger number of participants. Each user can select a preferred location in the mood space, which remains registered for a certain period of time until it expires. This creates a continuously changing average mood that is displayed to users by the circular player cursor as illustrated in Figure 4. The preference can be changed at any time, although if done while the last preference is still active, it is overwritten by the new selection, i.e. each user can only have one preference at a time.

For users who would like to explore the player on their own or with a select group of friends, we have introduced the notion of a party, somewhat similarly to the concept of a room in Dubtrack.fm. Every user who arrives at the front page is immediately added to the global Moodplay party where they can see the votes of all other participants. However, they are also able to create their own personal party and share the link to it with whomever they choose. All members of a party are able to vote on their select private musical sequence and always hear the same track at the same time, mixed together in the same way, regardless of where they are geographically. If they are in the same room, the users can decide to have only one of their devices play the music while still all of them can vote on silent devices.

The synchronization between different audience members of a party is ensure by socket.io[11] - a JavaScript library that enables real-time, bidirectional and event-based communication between clients and the server. A server gathers votes and push updates to all participants in a party at a specific interval which can be configured separately for every party. The updates contain information about currently active users, the average mood of the party, and information about the track that is playing. If the system detects that there are no currently active participants, it activates

[9]https://www.dubtrack.fm/
[10]https://www.playlist.com/
[11]https://socket.io

a number of bots that generate automatic votes to keep the party going. Interacting with other participants by selecting music according to mood is one way moodplay.github.io encourages social interaction, but it also facilitates sharing music between participants.

4. MUSIC SHARING

Users can also choose to add their own music that is not part of the official moodplay.github.io repertoire for use at their private party. They can simply drag and drop audio files onto the browser window. Using a statistical method a custom track's mood can be automatically inferred from audio features, which can be extracted directly in the browser using the framework *piper-js*.[12]

Users who want to share their music can use the mood plane to visualize of the music they are sharing. We explore two options. The first is to attribute mood variables valence and arousal in an automated way, using features extracted with vamp plugins, and the mood tags by humans for the existing dataset. We use various statistics of both high and low level features, like moments and autocorrelation. We trained a random forest classifier on these feature vectors to predict valence and arousal, and identify the features with the highest importances as the informative features for valence or arousal respectively. With those, we construct a distance measure between tracks. This distance is uses only extracted features, so it can be used to new uploaded tracks without mood tag. We use it to calculate the nearest neighbours of a track, separately for arousal and valence. The averaged mood values of the neighbours are then used as valence and arousal tag for an uploaded track. This approach works well however has some limitations. One is to choose the distance measure: Since the feature vector by which each song is represented has 400 dimensions or more, the problem of hubness is present (i.e. song songs have a low distance to many others because of the aggregation over many dimensions, but not because of apparent audio similarity). To reduce hubness, we reduce the number of dimensions to the ones of the most informative features (separately for valence and arousal). In addition we use a method from [4] that constrains the number of neighbours of hubs to the mutually closest neighbours. The attributed mood variables depend on these choices, as well as on the used data.

A second option is to let the user tag the uploaded song himself. This can be complementary to the first method, or be assisted by the first method, such that the system suggests mood coordinates, which the user can then accept, or modify. Even if it is not possible to place a track with high precision, it is useful to give a quick first description of the music a use wants to share. Another possibility is to use automated mood tags as suggestion which region of the mood plane to explore, and to find tracks with similar mood to an uploaded track, without the need to describe it or attribute genre tags to it.

5. AUTOMATIC MIXING

One of the advantages of recent developments in Web technologies and especially the Web Audio API is that the ways in which audio content can be presented to users online are much more varied than they used to be. Instead of simply playing back music files linearly, one can now easily create

[12]https://github.com/piper-audio

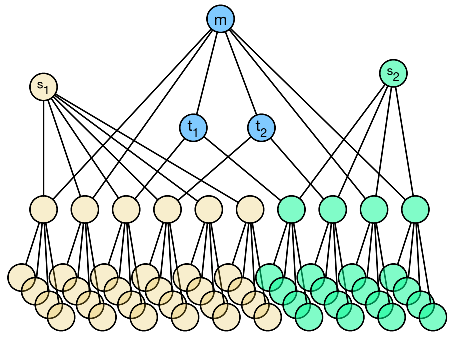

Figure 5: A simplified representation of a sample mix m from one piece s_1 to the next s_2, where the mix features bars from both pieces as well as combined transition parts t_1, t_2.

interactive applications with advanced audio functionality.[13] Streaming services can greatly benefit from these advancements and incorporate custom music processing functionality that is tailored to individual listeners.

Moodplay.github.io demonstrates this in the form of automatic DJ mixing which consists in creating smooth and appropriate transitions between subsequent songs for individual listeners or parties. Depending on the compatibility of two subsequent songs and their tempo, harmonic, and rhythmic content one of several available customizable transitions is chosen and adapted to the specific musical situation. This functionality is based on a Node.js module that can be embedded into any Web application.[14] It uses the Dynamic Music Objects framework to build DJ mixes on the fly based on features extracted from the audio[5]. The core of the package uses a decision tree to determine the best way to transition from one given song to the next depending on their degree of compatibility which is inferred via high-level analytical descriptors derived from the audio features. These decision trees can be custom-defined or they can be learned automatically from transitions ratings by users[5]. A root object of type sequence represents the whole mix to which the parts of the pieces intended to play are then gradually added. These parts can be of various types and may vary depending on the type of transition decided on. Figure 5 shows a simplified representation of a mix containing parts of two pieces. Depending on which root object is passed to the player, one can get it to either play the individual original songs, or the created mix, or all at the same time.

The auto-dj node module has a simple interface with a few public functions: `isReady()` which returns a Promise that resolves once the module is initialized, `getBeatObservable()` which returns an RxJS Observable which emits an incremented number whenever a beat is played (can for example be used for animations), `getTransitionObservable()` through which one can obtain an Observable that emits whenever a transition is started, `transitionToSong(audioUri: string)` which transitions to the song at the uri passed as an argument, and `playDjSet(audioUris: string[])` which mixes a whole set of songs. Anything else is done internally and automati-

[13]see for example https://tonejs.github.io/demos
[14]https://www.npmjs.com/package/auto-dj

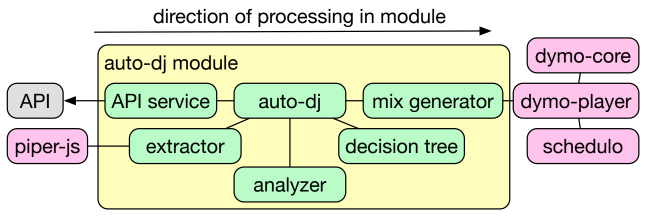

Figure 6: The current structure of the mixing functionality of moodplay.github.io.

cally, including loading the audio, extracting features, calculating higher-level descriptors, deciding on a transition, gradually adding to the mix structure, and playing it back. moodplay.github.io also makes use of the option of providing a custom feature service when initializing auto-dj, which provide pre-extracted features for all songs in the standard collection which significantly reduces browser load. For any custom songs (Section 4) a *feature extractor* extracts audio features directly in the user's browser using *piper-js*.

Figure 6 shows the structure of the current version of the automatic mixing module. An *analysis unit* calculates and buffers the high-level descriptors, a *decision unit* is capable of performing various decisions, and a *mix generator* contains templates for all the transition types, adds hierarchical song structures to the *dymo-core* triple store, and creates the mix object as described above. This object is then navigated by the *dymo-player*,[15] a playback module optimized with Web Workers and based on the dynamic scheduling module *schedulo* built around Tone.js[16] or *web-audio-scheduler*[17].

6. SUMMARY

moodplay.github.io is an online music streaming platform that strives to encourage social interactions in music streaming by allowing users to collaboratively choose music according to mood. Since it is a web application, it is accessible on any platform or operating system that supports web browsing. Users can participate in the global Moodplay party where everyone is added upon arrival, but they can also create personal parties and control who are invited. There is functionality that allows participants to share their favourite tracks with other invited participants. The system analyses the uploaded tracks by audio features to find their corresponding mood coordinates, so that the automatic DJ module can incorporate the new additions to the continuous mix. One of the many challenges we have faced implementing and deploying this system has been managing and optimizing the audio processing with the Web Audio API. Apart from trying to keep the CPU load at reasonable levels, we have encountered interesting dilemmas regarding where the different phases of processing should happen. Currently the audio is accessed directly from Deezer API and the server only deals with track metadata, user and party coordinates and determining which track should be mixed in next, leaving all the audio processing to the client devices. Alternatively, given sufficient server resources, the audio mixing could also take place on the server, that sends out the same stream to all the clients.

[15]https://github.com/dynamic-music/dymo-player
[16]https://tonejs.github.io
[17]https://www.npmjs.com/package/web-audio-scheduler

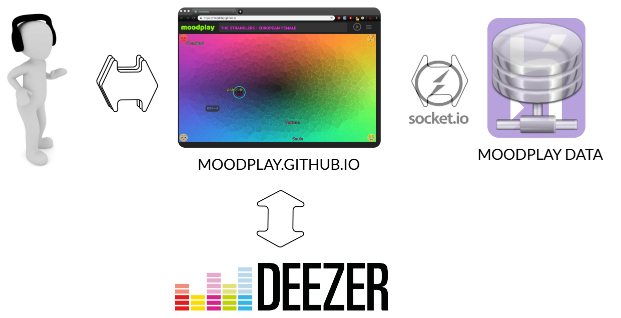

Figure 7: The structure of moodplay.github.io

Technically, the system consists of an Angular[18] front-end accessible at https://moodplay.github.io and Express.js[19] server application, https://moodplay-data.herokuapp.com/ that stores track metadata, mood coordinates and audio features for the auto-dj module. Both components are developed as open source projects under GNU General Public License v3.0. The code repository for the front-end can be accessed at https://github.com/darkjazz/moodplay and the back-end is available at https://github.com/darkjazz/moodplay-server.

7. ACKNOWLEDGMENTS

This work was supported by EPSRC Grant EP/L019981/1, “Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption”. We would like to acknowledge the contribution of Mathieu Barthet and György Fazekas, who created the original installation version of Moodplay.

8. REFERENCES

[1] A. Allik, G. Fazekas, M. Barthet, and M. Sandler. myMoodplay: an interactive mood-based music discovery app. In *Proc. of the 2nd Web Audio Conference (WAC)*, 2016.

[2] I. Andjelkovic, D. Parra, and J. O'Donovan. Moodplay: Interactive mood-based music discovery and recommendation. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, UMAP '16, pages 275–279, New York, NY, USA, 2016. ACM.

[3] M. Barthet, G. Fazekas, A. Allik, and M. B. Sandler. Moodplay: an interactive mood-based musical experience. In *Proceedings of the Audio Mostly 2015 on Interaction With Sound, AM '15, Thessaloniki, Greece, October 7-9, 2015*, pages 3:1–3:8, 2015.

[4] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer. Using mutual proximity to improve content-based audio similarity. In *ISMIR*, volume 11, pages 79–84, 2011.

[5] F. Thalmann, L. Thompson, and M. Sandler. A user-adaptive automated dj web app with object-based audio and crowd-sourced decision trees. In *Proceedings of the 4th Web Audio Conference, Berlin*, 2018.

[18]https://angular.io
[19]http://expressjs.com