Project Jukebox:
Rediscover Your Music Library

Kyle Dumovic
Computer Science Dept.
Stanford University
kdumovic@cs.stanford.edu

ABSTRACT
In this paper we describe Project Jukebox as a novel tool for exploring and visualizing the contents of a user's Spotify music library. Project Jukebox is an AngularJS web application that leverages the Spotify Web API to pull saved and top tracks from a user's library and visualizes them as nodes within a zoomable, multi-dimensional scatterplot built using D3.js. This visualization plots songs against the dates they were added to the library, in addition to audio feature set data provided by Spotify that classifies tracks according to their danceability, energy, speechiness, and more. Project Jukebox integrates with the YouTube Data API so that any song in the visualization can be played in its full length. This functionality allows users to rediscover tracks within their music libraries by providing the tools needed to browse their content in a visual way, explore trends and select songs based on their musical attributes all without the text-heavy interfaces so prevalent today.

Author Keywords
Spotify, D3.js, YouTube, AngularJS, information visualization, music discovery.

INTRODUCTION
Music streaming services like Spotify and Apple Music emphasize discovering new music that best suit a user's unique taste; however, few if any of these services offer tools to explore users' existing music libraries. Music libraries and playlists are typically displayed in boring, monochrome table layouts, supporting only searching and sorting to parse through. We believe that rediscovering tracks within your music library can be as valuable as discovering new ones. For example, users may derive more pleasure listening to their favorite songs from years past, tracks they might not have heard in a while and have likely forgotten about, more so than some new release that Spotify has suggested or the overplayed tracks on their Recently Added playlist. We argue that existing music management software like iTunes, Spotify and Google Play Music do not provide the tools for intelligently traversing and exploring existing music libraries—at least not in an intuitive, visual way. Project Jukebox aims to tackle this problem by providing the tools needed to extract and visualize the contents of any user's Spotify music library. It does this by leveraging the Spotify Web API to pull all the saved and top tracks from a user's library and visualizes them as nodes within a zoomable, multi-dimensional scatterplot built using D3.js. Tracks are displayed based on the dates they were added to the user's library, in addition to metrics that classify each track's musicality based on a set of audio features provided by Spotify including danceability, energy, speechiness, and so on. Project Jukebox also integrates with the YouTube Data API so that any song that has a music video uploaded to YouTube can be played in its full length. We have found that the feature set provided by Project Jukebox offers tools to browse and explore a Spotify music library in a way that does not exist today and in doing so provides value to its users.

RELATED WORK
Existing research has looked into new methods for music library navigation that does not rely on track metadata or tag information and is instead based on audio content [1]. Mueller et al. examined automated metrics for exploring music libraries based on content analysis and built a graph-based visual interface that leveraged spectrograms and a number of different layout algorithms. For our purposes, tracks' metadata is entirely present and properly curated by Spotify—a considerable improvement since the days of untagged or mistagged music prolific within users' hodgepodge music libraries. Additionally, Spotify earlier this year released new API endpoints making available individual track audio features: a dozen high-level acoustic attributes from the audio that are the result of a suite of audio analysis algorithms run on every track in Spotify's catalog [2]. This is the data that Spotify uses for its personalized recommendations. This makes it possible for us to harvest musical features like speechiness and danceability from users' tracks, which we display on multivariate axis, as described in detail later on.

Other research has looked into tree-maps, rectangle and disc visualizations as means of better organizing especially large collections of music [3]. Torrens et al. found each of these visualization techniques were well suited to purposes around giving an overview of music library contents, visualizing playlists and supporting users in their management and organization. For example, a disc visualization provided insight into the approximate size and typology of a newly created playlist, helping users find a

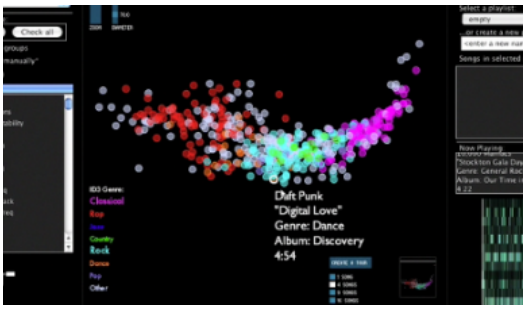

Figure 1. MusicBox has a user interface similar to that of Project Jukebox.

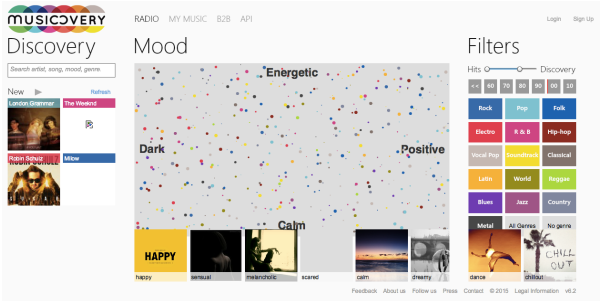

Figure 2. Musicovery has a four-way scatterplot based on tracks' moods.

better idea which "zones" of a library included in that playlist were overused or underused.

MusicBox, Anita Lillie's Masters Thesis at the MIT Media Lab, uses APIs built by The Echo Nest, acquired by Spotify two years ago, and the precursor to the audio features API. MusicBox was focused on the problem of finding music you like (e.g. by matching a particular mood) [4]. Its interface as shown in Figure 1 resembles that of Project Jukebox with its colored nodes and tooltip popovers. Similarly, Musicovery as shown in Figure 2 is a music discovery tool built in D3.js that utilizes internet radio and maps songs, colored by genre, along a four-way scatterplot with alternate ends representing opposite moods (e.g. energetic and calm, dark and positive) [5]. Project Jukebox derives some inspiration from this tool.

METHODS

Project Jukebox is an AngularJS web application built off the scaffolding of Jukebox-web, the web component of a collaborative playlist app for iOS built for CS 194: Software Project and CS 194H: User Interface Design Project. Project Jukebox is composed of a series of controllers and views. The `dataviz` controller corresponds to the `/dataviz` route at which Project Jukebox is located. Dataviz takes care of pulling user data from Spotify using the Spotify Web API.

Spotify Web API

We use the Angular-Spotify service [6] to connect to the Spotify Web API, make queries and authorize the Spotify login. Upon logging in, users must give Project Jukebox access to the following scopes: `user-library-read` and `user-top-read`. Note that write access is currently not necessary. The scope user-library-read gives us access to to the `/me/tracks/` API endpoint which returns a list of all a user's saved tracks. Note that this requires a user to have explicitly saved a track to their library by clicking the plus icon in Spotify. As it turns out, many users do not use this feature and instead add tracks to playlists directly. As a result, some users may find that Project Jukebox does not return as many tracks as expected. To resolve this, future work may query for all a user's playlists and add each unique track from each playlist to our data set. Also note that the `/me/tracks/` endpoint only returns 50 tracks with each query. It accepts an offset parameter so that we can query for more. Our implementation uses a recursive series of chained Javascript promises to make AJAX requests to this endpoint.

Unfortunately the saved tracks data is not personalized on a per-user basis beyond the inclusion of the date and time, an ISO-formatted string. This means that a song's play-count, —a metric we were excited to use for this project—is missing. Interestingly, so is genre. Spotify associates genres with albums, not songs. We could query for each track's album to gather this missing information, but that was more queries than we'd like. To make up for our missing play-count, we take advantage of the `/me/top/tracks/` endpoint, which returns a list of a user's most popular tracks from the last 4 weeks, last 6 months and of all time. Spotify does not reveal how this list is calculated. We iterate over each song in saved tracks and if it also appears as a top track for one of these time periods we make note of this in our master list object by adding an attribute.

Key	Value
acousticness	0.119
danceability	0.686
energy	0.754
instrumentalness	0
key	9
liveness	0.138
loudness	-4.537
speechiness	0.103
tempo	114.037
valence	0.737

Table 1. Audio feature data returned from Spotify associated with the track "I Know What You Did Last Summer" by Shawn Mendes.

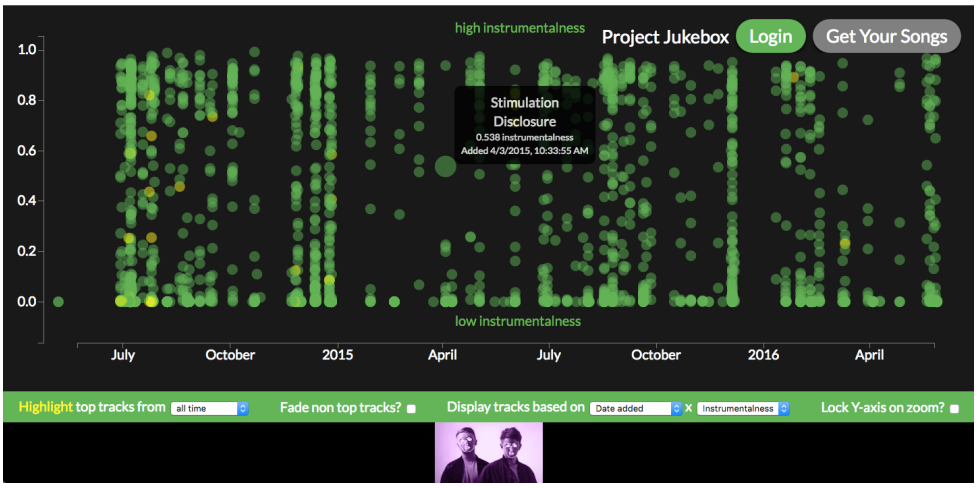

Figure 3. Project Jukebox in its final state. Here tracks' instrumentalness is mapped over time.

Finally, we query for audio features data from the `/audio-features/` endpoint. We can supply the ids of up to 100 tracks and Spotify will return the audio feature data for each track. We request each of these asynchronously and our callback decrements a counter acting as a bonafide countdown latch so we know when we've completed all our queries. We then associate each track in our master list with its audio feature data based on the offset index of the id of the requested track. For those curious, the audio feature data appears as it does in Table 1.

D3.js

Now that we've constructed our data object, we use D3.js to visualize it. We use the `graph` controller to compute and render its content. We chose to draw a two-dimensional scatterplot, with the x-axis representing the time the track was added to the user's library and the y-axis representing one of the audio features associated with that track, as shown in Figure 3. The `d3.scale` and `d3.domain` (for calculating range) functions were critical here, especially when initializing an axis for time. Each track is plotted as a semi-transparent green node, and each is assigned the hover behavior of a tooltip describing that song's name, artist and relevant audio features.

We also implemented `d3.zoom` behavior that allows the viewer to scroll to zoom and click to pan over the area of the scatterplot. As the user zooms the axis are adjusted accordingly. This feature allows users to drill down to a particular song, added on a particular day. In fact the resolution is so high that you can view the order in which you added songs to your library on a particular date, over the course of a particular hour. We used SASS (Syntactically Awesome Stylesheets) and gulp to produce the compiled and prefixed CSS we used to style and animate our nodes. We also added click behavior as described below.

YouTube Data API

We decided to embed a YouTube video player inside an iframe within our application that integrates with the YouTube Data API so that any song that has a music video uploaded to YouTube can be played in its full length. Thus when any node is clicked, that song begins playing, as shown in Figure 4. This is a work-around as the Spotify Web API does not support song playback outside of a 30 second sample. The way we load the appropriate video is by sending the YouTube Data API a search query that is the track name of the clicked node concatenated with its artist name. We return the YouTube video associated with the first result and swap out the iframe's url with the new video id. This loads the video instantaneously. While this solution is not perfect (not all Spotify songs have associated YouTube videos, nor does the result always match the requested video) but in our experience it seems to work accurately the vast majority of the time!

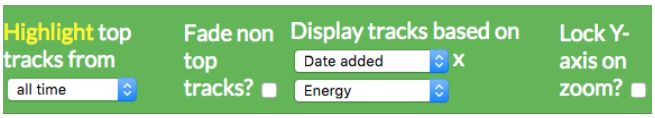

Figure 5. The control tools aren't particularly sophisticated, but swapping the variable axis can be powerful.

Controls

Consisting mostly of HTML form and input elements, we added controls to our D3 visualization to make it more interactive. These are handled by the `graph_controls` controller. As shown in Figure 5, the first control was the ability to highlight top tracks in yellow, by drop-down selection of one of the previously mentioned time periods. There is also a checkbox to toggle "fading out" non-top

Figure 4. A YouTube embedded player is fixed to the bottom of the application and allows any selected song to be played in full.

tracks, causing top tracks to stand out considerably more. We also added a toggle to "lock" the y-axis (representing the date and time) when zooming, as without locking the scale of, say, energy in each song, it isn't possible to drill-down to a daily or hourly view of songs added whilst keeping multiple nodes on the screen. Lastly, we added two drop-downs to select what each axis represents. Now, for

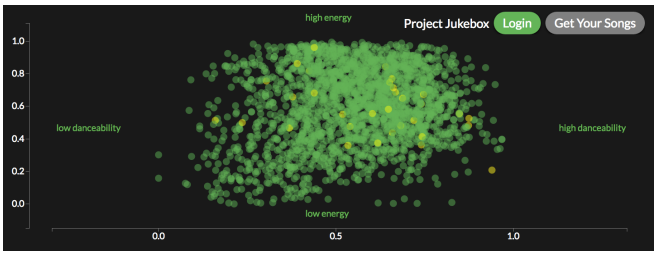

Figure 6. Project Jukebox keeps axis labels inside the content so that users do not need to constantly refer to the axis.

example, the x-axis representing time can instead represent danceability, and nodes can be viewed based how they map to both energy and danceability, like shown in Figure 6, or any combination of the audio features mentioned above.

Labels

Also shown in Figure 6, we added x and y-axis labels that sit beneath the plot itself instead of along the axis, indicating in this example which end of the graph is "low energy" vs "high energy." This takes cognitive load off of the user who now needn't constantly refer back to the axis.

RESULTS

Project Jukebox is capable of extracting the library information from a casual Spotify user's account and visualizing it in less than 3 seconds. For heavy users it can take significantly longer. Lawrence Rogers, one of our Beta users, managed to crash Project Jukebox because his Spotify library was so large. With over 2000 saved tracks, we would issue over 40 asynchronous requests, and Spotify would throttle us and reject every other one. We solved this by staggering queries with setTimeouts, but this significantly slowed our extraction process. Lawrence's Spotify account can take up to 15 seconds to load assuming it doesn't error out. For those curious to see how Project Jukebox handles such a large data set, we've made Lawrence's Spotify library available as the sample large data set that is accessible when you first load the application.

However, after loading these tracks, all computation is finished. Users are now free to interact with the scatterplot and associated controls. We have observed how they interact with Project Jukebox in many ways.

The "Oh yes, I remember that song"

Since Project Jukebox gives equal visual weight to any song in one's library (with the exception of top tracks, as we'll describe in a moment), users found it easy to discover or, rather, stumble upon, older songs that they might not have heard in a while. Users who signed into the app reacted positively, recognizing these as "their" songs. Unsurprisingly, users interacting with strangers' visualizations were less interested in the songs themselves. It turns out that many users don't actively remember their Spotify logins, so they were unable to use our tool and instead resorted to viewing a sample user's data set. As shown in Figure 7, with non-top tracks faded out it becomes even easier for users to see more of the songs they know and love. More often than not there was a "Yep, I listen to that song a lot! I love it!" reaction when users hovered over the yellow top tracks that stand out the most.

The "Yeah, that's me"

Another useful application of Project Jukebox is how it allows users to visualize their entire Spotify library in a single view. Whether they've added 100 saved tracks since creating their account, or 2000 saved tracks over the last couple years, users can view their libraries over a single time scale that lets them observe patterns in their listening behavior. Our users found that Project Jukebox helped them identify certain times in their lives when they were adding and consuming more music, and quiet periods when they

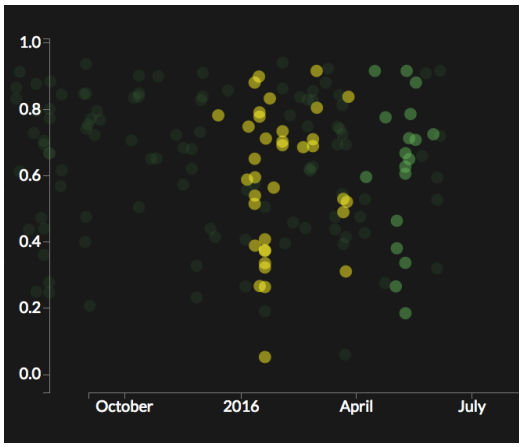

Figure 7. Note how easy to spot top tracks are when all other tracks are faded. Users enjoyed the splash of color highlighted top tracks provided.

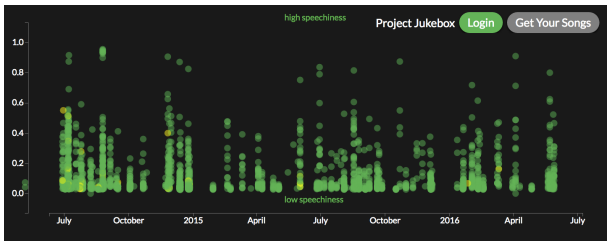

Figure 8. Note how the majority of this user's library has low speechiness, indicating that he mostly listens to background music.

were not. When toggling the y-axis from energy to a less evenly distributed metric like liveness, a user could see at a glance that their library just didn't consist of a lot of live songs. These users would then defend these characteristics, saying that this model rightly characterized them and their listening habits. See Figure 8.

The "Wow, that's cool"

We had so many users point out the novelty of the visualization tool. They especially liked when songs were mapped to danceability on one axis and energy on another. Users would try to guess which songs in their library would have high energy and high danceability, and more often than not they would agree wholeheartedly with the answer. On the opposite end, what were the songs that were low energy and difficult to dance to? For them, the visualization gave their data a fun twist. Users were especially impressed by the YouTube feature and enjoyed clicking on songs to give them a listen. This proved especially useful when there were tracks whose titles or artists users did not recognize until they were heard.

DISCUSSION

Project Jukebox was received remarkably well—indicating that users do in fact value rediscovering their own music rather than always listening to the same thing or giving in to the temptation of listening to something new. Visualizing something very personal like someone's own music library, and supporting a login-feature so users can work with their own data on their terms, inspire more of an attachment of users with our tool. If all datasets could be similarly and sufficiently personalized, information visualization would likely be a more popular field than it is today. Users were especially impressed by seeing their music library classified into audio features, and felt that the scatterplot was an effective tool for viewing these, though many did request that other scales be integrated, such as color, to bring additional information to each graph. The zooming as a traversal method was also particularly well-received, indicating that more visualizations today could take advantage of this feature.

FUTURE WORK

There are many directions in which Project Jukebox could go. The more interesting, we believe, are additional D3 visualizations that would take advantage of the substantial and personalized data set. For instance, tree-maps where tracks are categorized and grouped by genre, or force-directed graphs that identify playlist topology and library reach. Polar charts would also be an effective application of the audio feature data we've extracted.

There are also many ideas for possible extensions that could transform Project Jukebox from something that is fun to use once into something that can be used repeatedly and value can be derived from. Project Jukebox could allow rediscovered selected nodes to be grouped and playlists created that would then be synced back into Spotify. Search could be baked into the tool, allowing users to find specific tracks instead of just stumbling upon them, as well as filter what can sometimes be an overwhelming number of nodes.

As mentioned earlier, we would like to pull in tracks from all of a user's saved playlists rather than merely saved tracks, so that we can support the types of users that do not use Spotify's saved tracks feature and build a more accurate representation of their library. We also observed in testing that oftentimes not just one track is saved but an entire album is saved instead. Grouping nodes within an album and then expanding on zoom would make graphs significantly cleaner and more patterns could then emerge. We would like to see genre incorporated into our tool as well, perhaps reflected in colors, but this requires more work to fleshing out our data set.

ACKNOWLEDGMENTS

Special thanks to Lawrence Rogers for providing us with the login details to his Spotify account so that we could stress-test our app against a large dataset.

REFERENCES

1. Muelder, C., Provan, T., Ma K.L. Content Based Graph Visualizations of Audio Data for Music Library Navigation. *Ext. IEEE International Symposium on Multimedia, 2010.*
2. New Endpoints: Audio Features, Recommendations and User Taste - Spotify Developer https://developer.spotify.com/news-stories/2016/03/29/audio-features-recommendations-user-taste/
3. Torrens, M., Hertzog, P., Arcos J.L. Visualizing and Exploring Personal Music Libraries. *5th International Conference on Music Information Retrieval, Ext. ISMIR 2004.*
4. Lillie, A. MusicBox: Navigating the space of your music. *Program in Media Arts and Sciences, School of Architecture and Planning at MIT Media Lab, 2008.*
5. Musicovery, 2011. http://musicovery.com/
6. Angular-Spotify, GitHub. https://github.com/eddiemoore/angular-spotify