Seven Degrees of ~~Taylor Swift~~[1] Spotify: An Exploration of Artist Connections on Spotify

CS171 Final Project Proposal, Revised

April 18th, 2015

Angela Jiang | Jason Shen | Kit Wu

[1] We say this as a running joke, since Taylor Swift is unfortunately not on Spotify. If you search Taylor Swift in our initial start screen, you're out of luck – she won't show up. I think.

BACKGROUND AND MOTIVATION

We initially tried to do a final project on HIV/AIDS based on World Bank data, but quickly realized that the World Bank data was pretty spotty with a lot of countries. We consequently pivoted to another topic that was relevant to all of us: music.

Looking at Spotify, which all three of us use on a regular basis, we found an opportunity to visualize something that we're all familiar with and has relevance to a lot more college students than, say, HIV/AIDS (which, to be clear, is of utmost importance in the realm of global health – it's just not something that we think of every day). So with the data available to us through the Spotify API, we started thinking about what we can focus on and display in a fun, interactive manner.

PROJECT OBJECTIVES

The primary questions we are trying to answer are as followed:

1. Of all the information Spotify provides to us, what kind of info is the most suitable for visualization? What subject strikes a balance between feasibility and relevance to users? We eventually settled on visualizing the connections between artists, which is both feasible (as we already have a framework for creating nodes + links and manipulating them) and quite interesting.
2. What do we want to show through this visualization? At the moment, I see two options: we can either display/show high-level patterns between large amounts of different artists, or we can create something more interactive where the user can jump from one related artist to the next. The latter is more conducive to music exploration, if we implement the capability to play music from selected artists/nodes; the

former is more conducive to spotting patterns and larger trends. (As of Saturday night, 4/18/2015, it seems like we're leaning towards music exploration.)

3. How do users interact with the interface, and more fundamentally, how do users explore new music? If we are going to facilitate music exploration, we'll need to understand how Spotify users try out and listen to new music. We'll also need to understand how users interact with visualizations like the one we're proposing.

Through this project, we hope to:

1. Facilitate music exploration in a more visual, interactive manner than is currently done through the Spotify desktop client.
2. Highlight connections between artists that might otherwise go unseen.
3. Provide distraction to CS171 students and the larger Harvard community during reading period, or as they study for exams.
4. Show, on a high level, perhaps, connections between genres or larger trends in music.

DATA

We have successfully obtained the data we need to create a node-link visualization of a single artist and all related artists up to three degrees away through the use of the Spotify API. In our code, we're sending get requests to Spotify and upon receiving JSON files in response, we aggregate all the necessary information into two arrays of nodes and links (reminiscent of the Les Miserables file). Currently, we've been able to create nodes for hundreds of artists with no repeated artists (i.e. only a single node is created for a particular artist, even if that artist is related to multiple artists shown on the visualization), but a concern that we have is if we'll need to load new artists

every time a user selects an artist node as the related artists to *that* artist will be different from what we be had before.

If this sounds confusing, I understand. It's a lot easier when you can see the actual thing.

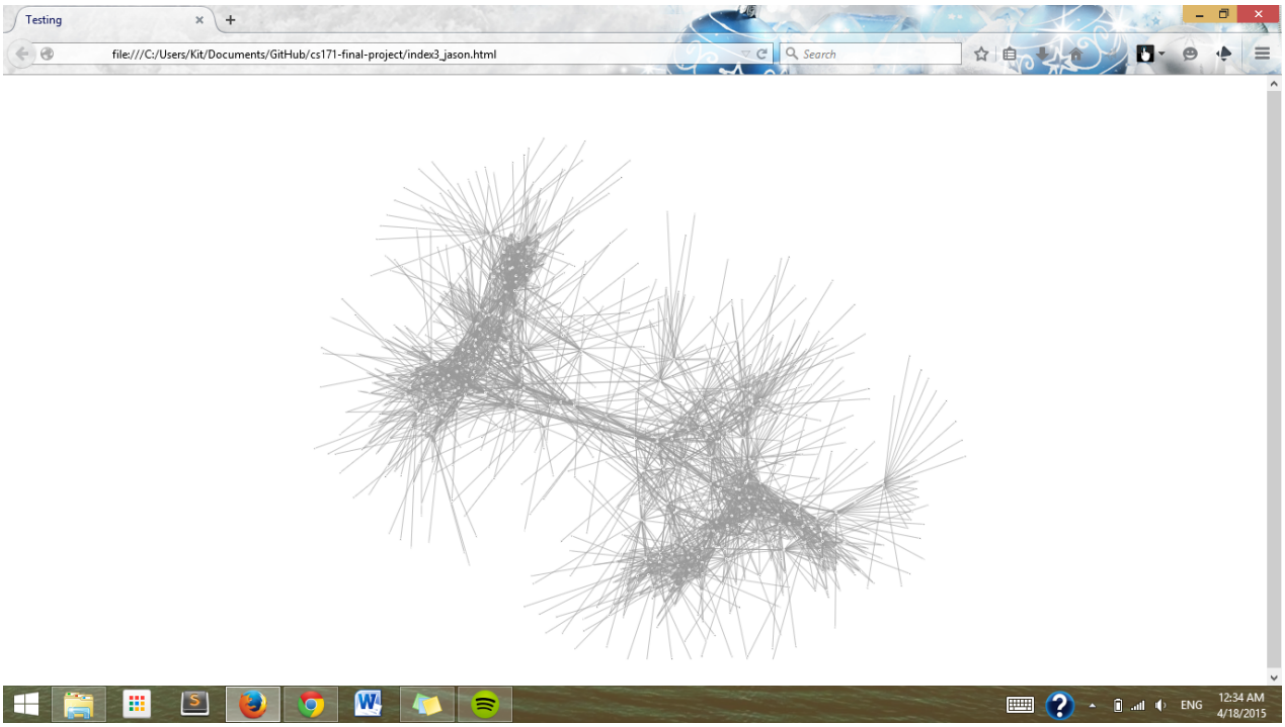
DATA PROCESSING

We've already managed to load the JSON files received from Spotify into arrays, and we've also managed to create links that connect the nodes which represent artists. For our most basic needs, data processing is not a concern. It might be a concern, however, as we implement update functions that involve the main/central artists changing.

VISUALIZATION

The easiest way to give you a sense of how our visualization is going to be is to go here: https://artistexplorer.spotify.com/. Our visualization aims to provide the same sense of music exploration, but with a force layout and interconnections between artists on all locations of the visualization.

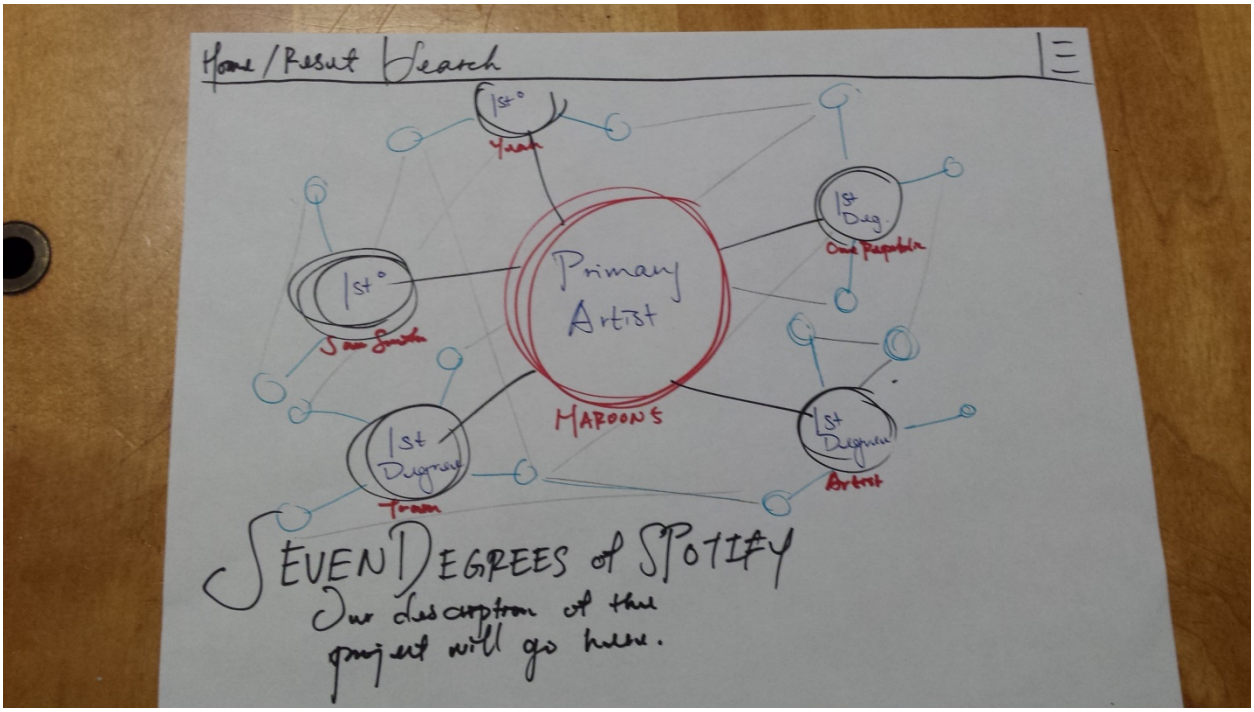
Currently, here's what we have in terms of nodes and links.



There's no differentiation between first degree, second degree, and third degree artists, but we'll implement those functionalities later.

(Note: Radiuses of nodes have been changed to a unit of one, so the nodes are essentially invisible. The links, however, are shown – to interesting effect.)

Here's a hand-drawn representation of how the final visualization might look like:



MUST HAVE FEATURES

- Nodes. We obviously want the artists – each of them, with no repeats – to show up as nodes in our visualization.
- Links. We need links between artists that are related to each other.
- An initial screen that prompts the user to enter an artist. This entry, in turn, will give us the artist ID that we'll use to grab data from the Spotify API.
- Size differentiation between primary/first/second/third degree artists. The first will be the largest, followed by the second, then the third.
- Solid update functions that allow for the user to jump from one artist to another. By this, we mean that the user will start off with a primary artist of, say, Maroon 5, **but will also have the ability to click a related artist (ex: OneRepublic) and have that artist become the primary artist.** This is going to be essential; without it, artist (and music) exploration cannot take place.

- Music playing function. We need to give the user the ability to click on a node, and listen to perhaps a sample of that artist's music. This will facilitate music discovery as the user travels from one known artist to the next, all the while sampling the artists they don't know about.

OPTIONAL FEATURES

- Differentiation between single-click and double-click commands, as in "single-click will play music, while double-click will jump to the selected artist." The reasons for doing this is based on user behavior; I've watched friends play around with the existing Artist Visualization that I've linked above, and something they mentioned was really annoying was the fact that songs switch every time you hover over a node. That means that songs will usually get cut off, unless you're willing to just not move your mouse. It would also be annoying, however, if the primary artist changes every time you click for a song; therefore, you need differentiation between playing music and updating nodes.
- Colors of nodes based on genre. Perhaps this will be the color of the node's perimeter (the image in the center will most likely be the artist's image on Spotify), but it would be nice to see differentiation based on gender (especially if we have a larger, more zoomed-out view like the screenshot I posted).
- Opacity based on degree. Would help with the ease of using this visualization; anything that helps the user focus frees up attention for the user to really explore the visualization and jump from one artist to the next. Moreover, given the visual nature of our artist exploration process, large amounts of nodes will be incredibly messy; we'll need to rein that in, so that the tradeoff for interactivity (organization) isn't completely sacrificed.

- Speed of transition. We still want a transition, but the faster users can sample music and move from one artist to another, the better their experience will be. Why do people use Spotify to stream music instead of Youtube? Youtube is free, but it takes you like, 15 seconds to go from song to song. You need to wait for the entire webpage to reload, and maybe you'll even run into an ad. On Spotify, however, the speed at which you can go from song to song is about a second. *Moreover*, Youtube does not display tracks all on a list for you; Spotify does. Spotify specializes in all the things that Youtube doesn't specialize in (it doesn't have to, as its primary value is in the display of video); if we design our visualization to emphasize those things, we'll hopefully create a Spotify-like experience as the user explores our project.

PROJECT SCHEDULE

It is currently April 18th, 2015. I'm going to go from here, as it is, after all, a revised project proposal.

By 4/25, Saturday | We'll have not only the nodes and links displayed, but also size differentiation between first/second/third degree nodes. We're also aiming to implement the update function that will give us artist-to-artist jump capabilities.

By 4/29, Wednesday | We'll have nodes and links displayed, size differentiation based on degree, artist-to-artist jump capabilities, names of artists displayed alongside nodes, hopefully pictures of the artists showing up in the nodes and perhaps music playing upon click as well. *Given that a lot of stuff happens during reading week, we're going to try to implement as many features as possible by this date so as to give us leeway before the final May 5th due date.*

By 5/3, Sunday | We'll have all of those things described above, and we'll implement the starting screen that requests for your artist, a search bar

on top, a reset view (possibly) that allows you to zoom out and see everything with size of nodes constant for all degrees, a sidebar that perhaps maps your artist-to-artist jumps and lets you jump backward to any previous artists if you wish to do so, and single/double-click differentiation.

By 5/4, Monday | Final push; general debugging, finishing touches, CSS and font fixes.

5/5, Tuesday | Project due.

Up until this point, we have used a combination of working as a group (where all three of us are working in the same setting) and working in smaller groups (where two of us work together when we have time). We are not yet at the point where we can delegate modules to individuals at the moment; Angela is capable of coding on her own, but Jason and Kit find that working together is more beneficial in terms of both coding and design.

EXTRA: DESIGN STUDIO PROCESS

This section might or might not have images included, depending on whether one of our team members (who has the design sketches) chooses to upload the sketches or not. In either case, all the adequate info is described in the seven pages prior to this one.