INFO 5602 Final Project Writeup

Lawrence J. Hessburg*

University of Colorado Boulder

ABSTRACT

For the past 4 years, I have been using a service called Last.fm to record every song I listen to from my music streaming apps. In this time span, I have gathered roughly 45,000 individual listens (or "scrobbles"). The Last.fm website automatically provides some basic statistics about this dataset, along with some limited browsing features, but I wanted to gain some deeper insight into my music listening trends, and correlate them with major life events. To do this, I scraped my data from the service using the Last.fm web API, and processed the resulting dataset to produce some basic visualizations about my music listening habits, and how they have been affected by major events in my life over the past few years.

1 INTRODUCTION

Last.fm is a music website that was originally introduced in 2002 as a music recommendation service. Using an open-source API, users were able to create applications which recorded individual listens (called "scrobbles") from their music-listening platforms (such as Spotify, Soundcloud, iTunes, online radio stations, and even offline apps like Windows Media Player) and uploaded them to the Last.fm database. By gathering a dataset of the music-listening habits of each individual user, Last.fm aimed to provide quality recommendations for new music to its users by linking users with similar taste.

While the music-recommendation component of Last.fm's service never really took off, and indeed the platform itself is still not widespread in the mainstream, there has been a strong, steady userbase, and a growing community of developers contributing to the service since it's inception. As digital music consumption has moved from locally downloaded files to online streaming over the past decade, the community has continued to maintain support for recording scrobbles from modern streaming platforms. Most modern streaming platforms contain their own robust music-recommendation services that often use modern big-data and machine-learning techniques to provide useful recommendations, and in that regard Last.fm has fallen to the wayside. In recent years however, the platform has transitioned from being primarily a recommendation service to almost a sort of music-based social media service, driven by its users music listening data.

Within the last few years, Last.fm has implemented multiple ways to browse through your music-listening dataset through its web interface, and view users with tastes similar to yours. Additionally, the service has added a number of basic data visualizations to view your music listenind habits over time. Examples of some of these can be found below:

While browsing these pages is entertaining and somewhat insightful, the main focus of these visualizations is to associate and compare one's music listening trend's with other users on the platform, rather than providing deep insight into one's individual habits and trends over time.

Since each "scrobble" is annotated with the track name, album, artist, time and date of listen, and other useful metadata, the full set of one's scrobbles provides an incredibly rich dataset that has a staggering number of exploration options. With this project, my goal was to harness my own scrobble dataset and associate it with various major life events over the course of the last few years, to see if any interesting patters seem to emerge.

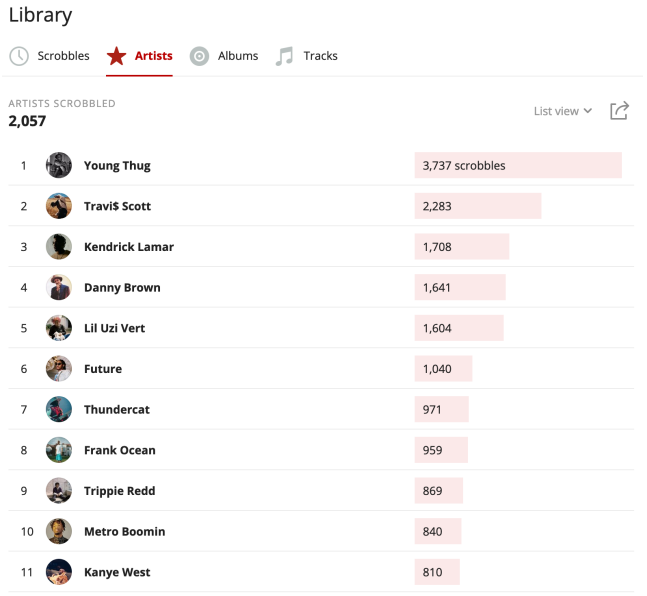

Figure 1: My top artists of all time as visualized by Last.fm

2 RELATED WORK

As mentioned previously, Last.fm[1] itself does provide a number of data visualization options. More specifically, it provides the ability to select any arbitrary range of dates, and view one's artists, albums, and tracks in order of most scrobbles within that date range. An example of this is found in Figure 1. Additionally, the service provides the Last.week, Last.month and Last.year pages, which provide a summary of one's scrobbling history over the past week, month and year respectively. These summaries include total number of artists, albums and tracks scrobbled in these timeframes as a percentile of the overall last.fm userbase. Also, they report histograms of scrobbling habits by time of day and day of week within the date ranges. Lastly, they provide a list of other Last.fm users with similar scrobbling reports within the date range.

Due to the robust and user-friendly web API provided by Last.fm[2], there are a plethora of user-developed visualizations of last.fm data. Generally, these visualizations provide time-series representations of specific aspects of the dataset such as stream graphs or bar charts, and other visualizations like word-clouds based on genre tags[3,5]. An example of one such user-made vis can be found in Figure 3.

Aside from visualizations, there are also a multitude of user-made applications that interface with the Last.fm web API to either scrobble listens to the dataset, or download records of scrobbles by

*e-mail: lahe2512@colorado.edu

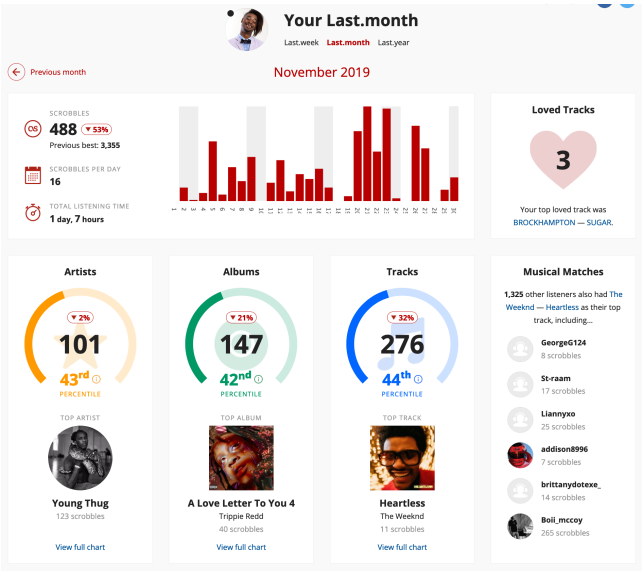

Figure 2: Some statistics about my last month of listening, along with similar users

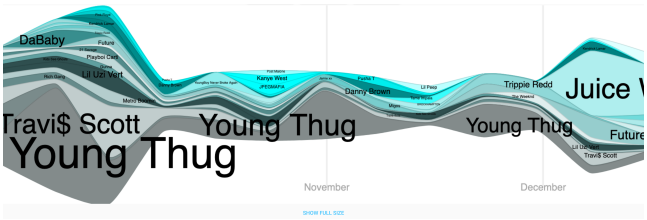

Figure 3: Stream graph of my top artists for the past 3 months as visualized by savas.ca/lastwave

user. There are even apps such as VinylScrobbler[7] which allow a user to simply enter the name of an album they are listening to in physical format, and scrobble said album's tracks to their Last.fm profile, timestamped appropriately for the length of said tracks.

Lastly, there are many more community-developed repositories that act as interfaces and/or wrappers between the Last.fm web API and common programming languages such as Python and Javascript[4].

To reduce repeated work, many of these projects were integrated into my own. Further details into the implementation can be found below.

3 Project Description

3.1 Data Acquisition

The first step in implementing my visualizations was collecting my scrobble data from the Last.fm database. I spent a significant amount of time researching various options. The first was to write a web scraper from scratch using my Last.fm account's API key. Since the Last.fm API conforms to the REST standard[11], this process is actually relatively straightforward (albeit time-consuming), and response payloads can be easily formatted and concatenated with common formats like XML and JSON. The second option was to use a pre-existing interface such as pylast to handle the API calls, and write the code for converting the response payloads into something more useable suh as a .csv file. Lastly, I discovered a user-made application called Last.fm to csv, which automates the entire process. This service only includes track, album, artist and time/date, excluding other metadata such as genre tags, which is sub optimal for my goal. However, due to time constraints I opted for this as my data acquisition method. Last.fm to csv provides a .csv file containing one line for every scrobble in my account's history, with the aforementioned parameters.

3.2 Data Preprocessing

The hardest and most time-consuming part of this project was the data preprocessing. I originally set out to visualize trends in properties like my top-n artists over time. However, converting a list of individual datetime-labeled scrobbles into a format usable for purposes like that turned out to be a very non-trivial data structures problem. This is due to the fact that I had to essentially move through time, keeping track of my top-n artists at the current time step in a separate data structure, along with other useful data I wanted to visualize. For all internal data representation and preprocessing, I used the common python libraries numpy and pandas.

3.3 Visualization Implementation

Once I had the data processing sorted out, I explored various ways to visualize said data. Because I had some experience with it from the other projects in this class, and becaus the end goal was to host the visualization on a web page, I opted to use the python library bokeh for my vis. Bokeh is a visualization library specifically targeted for web hosting, and provides a high degree of built-in interactivity that can be utilized. Additionally, it integrates quite seamlessly with other common python libraries such as numpy, pandas and matplotlib.

3.4 Visualization Features

The visualization I implemented has 3 main components, which are linked by their x axes. The first is a normalized stacked-area chart depicting the evolution of my top 6 artists of all time over since 1 January 2016. The chart represents my top 6 artists of all time at each day within the overall history of my dataset, as a percentage of my top 6 artists by scrobbles. I chose this as a starting date because my Last.fm scrobbles are very inconsistent before that ad I had not properly set up scrobbling services with my music streaming apps. Also, this gives a date range of almost exactly 3 years. My code allows for this same visualization for my top-n artists of all time, not just 6. The reason I chose 6 is because in total, I have had exactly 20 distinct artists in my all-time top 6 over the past 3 years. It was very difficult to find a categorical color palette with more than 20 categories that provided enough contrast between categories to distinguish between artists while also being somewhat aesthetically pleasing. After playing around with many different less-than-ideal options, I decided to reduce the number of artists in my stacked area chart instead.

The second main component of my vis is a line graph depicting the number of unique artists, albums and tracks scrobbled by week during this 3 year timespan. For this, I stepped through each week in my dataset, and recorded the total number of unique artists, albums and tracks listened to for the plot.

The third main component of the vis is another line graph depicting the evolution of the total number of unique artists, albums and tracks scrobbled since the beginning of my Last.fm history.

In addition to these 3 main components, the vis contains many other features. As stated above, the 3 visualizations are linked in their x axis. There is a selection tool displayed above the visualizations. Using your mouse, you can adjust the edges of the box within the selection tool, and drag the box horizontally to select a smaller window within the overall time series. All 3 visualizations will respond to this manipulation in real-time, and allow you to focus in on different time ranges within the overall time series.

One of the most important aspects of the visualization is the vertical gray lines that mark specific dates throughout each of the

3 time series. These lines represent major events in my life, and hovering over these lines within the stacked area chart will reveal a tooltip describing what event they represent. Some of these lines also represent album releases for albums that have significantly affected my overall listening stats within the past 3 years.

The line graph components of the visualization also contain tooltips. As you hover your mouse pointer over the line graphs, these tooltips will display the date, as well as values represented by the various lines.

4 Analysis

Although I was not able to achieve the level of depth I originally intended when coming up with this project, there is still some interesting insight to be gained from this visualization. Upon first inspection, there is one feature within the visualization that stands out far above the rest. Specifically, there are 3 distinct periods of extremely heightened music-listening activity, which is the most salient feature of the line graph depicting unique weekly scrobbles. Additionally, these 3 periods of activity all fall almost exactly within a specific span of two life event markers. Upon inspection, all 3 of these periods line up very closely with the time spans I was in Houston working at NASA as a Pathways Intern. This suggests that when I am working full-time, I tend to listen to much more music than I do when at school. Another interesting component to this feature lies in the third graph - the line graph representing total unique scrobbles. The steeper the slope of these lines, the more new music I am discovering, as the total number of unique artists, albums and scrobbles increases at a greater rate. The areas of steepest slope also line up exactly with the 3 aforementioned periods, suggesting that when I work full-time, not only do I listen to more music in general, but to more new music as well, whereas at school I tend to listen to most of the same artists and albums.

One thing that also stood out to me is the number of sharp discontinuities within the stacked area chart. Initially, I thought this might have been a result of my data preprocessing gone wrong, but after considering it some more I came upon a different realization. These discontinuities represent events in which my number 6 all time artist traded places with my number 7 all time artist. Upon first glance this does not seem very remarkable, however these discontinuities appeared no matter how many artists I chose for the stacked area chart. This means that the total number of scrobbles for my top artists over time declines very smoothly , and that (with the exception of the top 3 or 4 all time) each rank is very close in number of total scrobbles to the ranks before and after it. This means that my scrobbling history is not dominated by a small set of artists I listen to constantly, but the distribution is spread out a little more evenly, which is an interesting conclusion.

It is also interesting to note the vertical markers representing significant album releases within the represented time span. For example, the release of Kendrick Lamar's DAMN coincides with a very sharp increase in Kendrick Lamar's portion of the normalized stacked area chart (represented by the dark blue volume on the bottom), however, as time extends from the release of the album, his share of the plot steadily declines. It will bne interesting to recreate this visualization a few months after his next album is released to see if the pattern repeats.

5 Challenges and Future Work

5.1 Challenges

The biggest disappointment for me with this visualization is that I originally wanted a much higher degree of interactivity. You will notice quite quickly that there are no labels for the areas within the stacked area chart. This is because bokeh (as of the latest version 1.4.0) does not include functionality for hover tools within vertical area objects. Additionally, I was unable to get a simple legend to work as well because by default it would cover and obscure a large chunk of the stacked area chart, and moving it outside the plot itself breaks interactivity with the other visualizations. These are not problems with my implementation (messy as it may be) but inherent problems with bokeh itself. I spent way too long digging into github issues and forum posts trying to figure this out, and implement it myself before I eventually gave up and cut my losses. This is obviously a huge problem as the stacked area chart loses a lot of its meaning without these labels, but the silver lining is that the bokeh developers have included this exact functionality in their version 2.0 milestones, so it will get better in the future.

5.2 Future Work

Due to the limited time and resources I had to work on this project, I was not able to explore this dataset in nearly as deep a fashion as I indended. However, this is something I have wanted to do since quite a long time before even taking this class, and I do plan to use this as a starting point to expand upon during my free time in the coming months. Here is a basic list of functionality I wish to implement:

- Write custom web scraper with pylast that records additional metadata such as genre tags
- Add hovertools for stacked area chart once bokeh releases version 2.0
- Include dropdown menu to select top albums, artists or tracks for stacked area chart
- include slider to select number of top albums, artists and tracks to include in stacked area chart

Long term goals:

- Set up website to automatically scrape recent scrobbles and update visualization accordingly
- Add ability for other users to enter their last.fm username and generate a web page for their data!
- Add ability to tag major life events as they happen
- Explore the limitless other visualization possibilities from this dataset!

References

[1] Last.fm, https://www.last.fm/home
[2] Last.fm API reference, https://www.last.fm/api/
[3] LastWave, https://savas.ca/lastwave
[4] pylast, https://github.com/pylast/pylast
[5] Last.fm extra stats, http://c26k.com/lastfmextrastats/
[6] Last.fm to csv, https://benjaminbenben.com/lastfm-to-csv/
[7] Vinyl Scrobbler, https://vinylscrobbler.com/
[8] Bokeh, https://docs.bokeh.org/en/latest/index.html
[9] Pandas, https://pandas.pydata.org/
[10] Numpy, https://numpy.org/
[11] REST Architectural Constraints, https://restfulapi.net/rest-architectural-constraints/