Explorify: A Personalized Interactive Visualization Tool for Spotify Listening History

Inna Ivanova
University of British Columbia
innai@cs.ubc.ca

Jonatan Sissener Engstad
Norwegian University of Science and Technology
jonatane@stud.ntnu.no

Abstract

There are many music streaming platforms but Spotify has dominated the market and is currently the largest one with millions of users. Spotify collects huge amounts of data from their users on a daily basis which allows for a great research opportunities in the data science community. There are very limited attempts that have been published on the topic of visualizing Spotify streaming history which inspired this project. We introduce Explorify: a personalized interactive visualization tool for Spotify listening history. Explorify offers a plethora of interactive visualizations that help Spotify users explore their own music taste, patterns and enjoy the interaction with their own data.

1. Introduction

Spotify[1] is the largest music streaming platform with millions of users. Spotify has revolutionized music discovery by collecting streaming data from their users and applying modern neural algorithms for their music recommendations. In addition, Spotify provides information and audio features for each track and artist on the platform which is easily retrievable through their API[2]. Users are even able to request their own personal data including streaming history, playlists, top artists and others[3]. While there are a lot of visualization attempts for Spotify data on the Internet they are relatively simple and do not have interactive components. In this project we propose a novel visualization platform for personalized music streaming history provided by Spotify called Explorify. The main goal of Explorify is to allow users to visualize in an interactive manner their personal streaming history and discover patterns for their music taste.

[1]https://spotify.com
[2]https://developer.spotify.com/documentation/web-api/
[3]https://support.spotify.com/us/article/gdpr-article-15-information/

2. Related Work

Few publications about attempts at visualizing the listening history of Spotify users exist. More work exists on the topic of visualizing the listening histories of users of the Last.fm[4] platform, a service with very similar offerings to that of Spotify. There are differences in what data the two platforms make available, but they are mostly minor, and the semantics remain largely the same. In discussing these works, we will usually make no distinction between what platform the work relies on, as it (in most cases) is immaterial to the work.

The published works within this area can be divided into two groups, based on their main goal. The first group's goal is to help the user discover new music. The papers targeting this goal seek to create tools that reveal tracks, artists and genres that were previously unknown but are believed to be enjoyable to the user. Usually, this is achieved by allowing the user to explore and traverse networks of similar artists in search of undiscovered ones [1, 2]. Several tools that use this approach are available on the Internet. To name a few: LivePlasma[5], MusicRoamer[6], Music-graph[7], Music-map[8].

The goal of the second group is to reveal trends and facts about the user's listening history, supporting the user in introspection about their own listening habits. These tools all visualize the user's listening history in one or more ways, and support various means of interaction such as adjusting the granularity of what is shown [3, 4, 5]. Another tool, LastHistory, attempts to make this presentation more meaningful to the user by augmenting the listening history with contextual information from other data sources, such as personal phots and calendar entries [6].

In a recently published work, Wirfs-Brock et al. explore how these goals can be combined, enhancing music explo-

[4]https://last.fm
[5]https://liveplasma.com/
[6]https://musicroamer.com/
[7]https://music-graph.xyz/
[8]https://music-map.com/

ration by giving the user deeper insight into their own listening history and listening patterns [7].

While Explorify falls into the second group, we hope that our solution will facilitate discovery by piquing the user's interest and deepening their insight into their own music preferences in a way that will inspire exploration.

3. Dataset

There are a total of three streaming history datasets **SH_J, SH_D** and **SH_K** obtained from users **J, D** and **K** respectively, that were used during the development of Explorify. Each dataset is a timeseries dataset and comprises the streaming history for a user over one year period, in json format. Each item of the datasets contains:

- `trackName`: unique sting for each track, categorical data
- `artistName`: unique artist for the track, categorical data
- `msPlayed`: integer describing for how long the user has listened to the track in milliseconds, quantitative sequential data
- `endTime`: a timestamp describing the exact time at which the user stopped listening to the track, quantitative sequential data

A summary of dataset cardinalities is provided in Table 1.

	Total Tracks		**Unique Tracks**	**Unique Artists**
	Skipped	**Not Skipped**		
SH_J	1126	6887	4272	1598
SH_D	1919	3570	1428	375
SH_K	1862	1711	1451	818

Table 1. Datasets overview for each user **J, D** and **K**. Skipped tracks are ones that are played for less than 10 seconds.

By retrieving information from the Spotify API, the raw datasets are further processed to extract detailed information for each track and the corresponding artists and album. Data processing libraries such as `numpy`, `pandas` and `sklearn` are used for the computations. Examples of additional information provided by the Spotify API include:

- Track audio features which are all quantitative sequential data with different ranges:

  1. danceability
  2. valence
  3. energy
  4. tempo
  5. loudness
  6. speechiness
  7. instrumentalness
  8. liveness
  9. acousticness

- Genres associated with the artist: categorical data;
- Related artists of an artist: Artists which listeners of the current artists are likely to enjoy as well;
- Popularity score of the artist: ordinal data in the range [0, 100];
- Number of followers the artist has: quantitative sequential data;

Dataset availability is both static and dynamic. The static part is the streaming history provided by the user. The dynamic part is the prepossessing of the raw user data and the additional features fetched through the Spotify API.

4. Data and Task Abstractions

There are a total of 5 tasks we will be visualizing within the Explorify platform: **Artist-Genre Network**, **Track Clustering by Audio Features**, **Artists Streamgraph**, **Top Artist Over Time** and **Daily Listening Pattern**. While we wish to facilitate all of these tasks with Explorify, there are some which are more important than others. In particular, we will prioritize the artist-genre network and daily listening pattern tasks. The streamgraph and track clustering tasks on the other hand will only be implemented if we are happy with the implementation of the other tasks.

4.1. Artist-Genre Network

Spotify users often want to explore the relationships between the artists they listen to the most. They may want to know how genre connects and differentiates the artists they listen to and in particular to identify which genres connect two artists and how other genres belonging to only one artist make the pair differ. This task aims to allow the users to explore those relationships in an interactive way by visualizing the artist-genre network. Our hypothesis is that artists connected by genres will not be associated with other completely different genres. To verify this hypothesis, the user needs to consume the network graph and discover patterns and abnormalities.

Both **genre** and **artist** are categorical attributes. The cardinality of the artist data is shown in Table 1. The cardinality of the genre attribute ranges between 249 and 1049 with the different datasets. Each artist has a number of associated genres, linking the two attributes. This makes it possible to construct networks where artists are vertexes and genres are edges or the other way around. In one dataset, SH_J, each artist has 4.5 genres on average, and each genre links 6.5 artists on average. This means that the degree of connectivity in the network is quite high. Because the number of artists in each dataset can be in the order of thousands, we will be performing a filtering reduction, limiting the visualization to displaying only the user's top 100 artists. This will help achieve a meaningful visualization while avoiding the hairball issue [8]. In terms of abstract definition, this task explores the connectivity, and lack thereof, of nodes within a network.

4.1.1 Calendar-heatmap connectivity

The user can aid their search and/or browsing of the network by selecting days in the heatmap calendar (see Section 4.5). This action will highlight the artists which were streamed on the selected day in the artist-genre network.

4.2. Track Clustering by Audio Features

Spotify users have different listening patterns and they might listen to a variety of music styles. The goal of the second task is to allow the user to explore their music taste preferences and discover patterns by clustering user tracks according to the audio features. Questions this particular task aims to answer are:

- Which tracks are similar according to their audio features?
- What is the predominant genre within a cluster and is there a distinct winner?

There are a total of 9 audio features attributes for each track (see Sectiton 3). All these features are ordered quantitative sequential data. Each track is represented in a high dimensional space with its audio feature vector. Each track is further assigned a list of genres. There is a 1 to Many relationship between the track and the associated genres. The genre attribute is a categorical data. To obtain the data for this visualization, we first cluster the tracks into high dimensional space and then derive a representation into 2D space for each datapoint. We will use all tracks for this visualization to obtain reasonable results for the clustering. The cardinally of the track data is described in Table 1. Our hypothesis is that tracks with similar audio features will be mapped close together in the low dimensional space and will have similar genres assign to them. In terms of abstract definition, the task explores relationships between high dimensional data points within assigned cluster and searches for patterns.

4.3. Artist Streams Time-Series

In the third task, we wish to let the user explore the evolution of their listening patterns for the duration of their retrieved history. The goal is to allow the user to discover events such as when they first started listening to an artist, and patterns such as how the user's interest in an artist changed over time. We also wish to make it easy for the user to discover which artists they streamed the most, as well as the absolute number of streams at any point in time. The data used for this task will be the following:

- Artist id: categorical. Cardinality is documented in Table 1.
- Artist name: categorical. Same cardinality as artist id
- Timestamp: quantitative, sequential, hierarchical. Cardinality: 1 year, accurate down to the second.

Each data set covers around one year of a user's listening history, with the streams roughly spread evenly throughout the period. It follows that the range of the timestamp data is a single year. Because the timestamps are accurate down to the second, some aggregation will have to be performed so that items can be grouped together by temporal closeness. Grouping the items on a set number of days, like 5 or 7 will yield a good number of groups, being granular enough to reveal real detail without being so fine-grained that overarching patterns are hard to discern. There are no levels in the categorical data. Artist id should map 1-1 to artist name, though artist id will be used internally in the system to be safe.

As the cardinality of the the artist id attribute is very high in some of the data sets, we will be performing a filtering reduction. Only the 150 (or thereabouts) most popular artists will be included in the visualization.

For this task the data will be structured in the following way: For each set time-interval, every artist in the data set will be associated with the number of times they occur within that interval. One can then visualize, for each interval, the absolute or relative number of streams each artist received.

4.4. Top Artists Over Time

Spotify users often search their top artists through the Spotify platform. However, the visualization Spotify provides is very limited because it provides the user's top artist only over the last 4 weeks, 6 months or entire history. This restricts users to explore in details their most popular artists for specific time period and to track the artists popularity trends. The goal of this task is still to provide the user with the most streamed artists over the period of their **entire** history and to also allow them to look up their top artists for specific days. The data used for this task is very similar to the one used in task three:

- `artistName`: categorical data. Cardinality is documented in Table 1.
- `msPlayed`: quantitative sequential data. Cardinality is same as total tracks number and is documented in Table 1.
- `endTime`: quantitative sequential data. Cardinality is same as total tracks number and is documented in Table 1.

To obtain the listening times for all artists we will be aggregating the listening times (`msPlayed` attribute) for their tracks over all and specific dates (the `endTime` attribute).

Since the cardinality of the artist over all datasets is quite high (in the hundreds to thousands) we will be performing filtering reduction – only the top 20 artists will be displayed at any time. To obtain the actual top 20 artist we will perform data reordering (sorting). In terms of abstract definition, this task explores the distribution (interaction time) for discrete data (the artists) over a period of time.

4.5. Daily Listening Pattern

The final fifth task is visualizing the daily Spotify usage of a user. The goal of the task is to let users explore their daily streaming history and discover interesting patterns. Questions this particular task aims to answer are:

- How music streaming routine changes over time?
- Which days observe the most listening hours?
- Is there spike usage for specific dates?

The data used for this task will be:

- `msPlayed`: quantitative sequential data. Cardinality is same as total tracks number and is documented in Table 1.
- `endTime`: quantitative sequential data. Cardinality is same as total tracks number and is documented in Table 1.

To retrieve the data needed for visualization of the task, the listening interaction on a particular day are summed up together (aggregated). This is a time series dataset so it's easy to aggregate tracks times by day. This results in a new attribute `streaming_time` that is ordered quantitative sequential data. The cardinality of the resulting dataset is 365 since the original datasets contain information over a year and we aggregate that information by dates. In terms of abstract definition, this task displays the distribution of streaming time over days.

5. Solution

The work on the Explorify platform is performed using Python and JavaScript with D3 [9] and React [10]. Python is mainly used for data processing while JavaScript with D3 is used for creating the visualizations.

5.1. Artist-Genre Network: Force-Directed Graph or Adjacency Matrix

Since the data to be visualized for task one (Section 4.1) is a network type a total of two visualizations are considered: adjacency matrix presentation and force-directed graph inspired by a paper on a similar visualization task [11]. The preference is over the force-directed graph, however, depending on the time constrains and challenges around the implementation, we might end up with the simpler adjacency matrix. Whichever visualization we land on, we will aim to be interactive.

In the force-directed graph each artist will be encoded by a node and each genre connection by an edge. Since artists could be assigned more than one genre some edges might have higher 'weight'. Therefore, the edge width will encode the number of genres in common – the more the wider the line. The colour channel can be used for highlighting some connections of interests. To inspect specific artists the user will be allowed to click on the artist (node) of interest and compare to another artist that is a connection. A pop-up display will show detailed information about the similarity and difference in genres between the two artists. Current progress of the implementation shown in Figure 1.

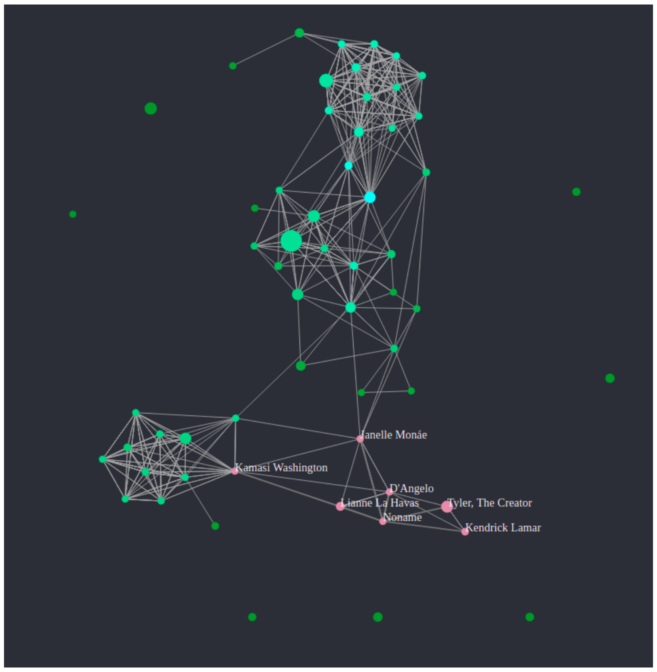

Figure 1. Artist-genre network visualization

An alternative to the force-directed graph is to create a variation of an EdgeMap [12]. With this idiom, we would have artists be nodes, and edges represent that two artists were streamed during the same session, with edge thickness being proportional to the number of session co-occurrences. The timeline view will position artists by the data on which they were first streamed. We are currently exploring this possibility.

In the case of adjacency matrix each row and column item will correspond to the unique artist. The values of the cells in the matrix will display the number of genres in common between two artists. To encode the number of equal genres the colour and hue channel will be used. We will work with a single colour sequential colour map because the number of genres in common is always a non-negative number. To inspect a connection of interest, the user will be allowed to click on specific cell. As with the force-directed graph, a pop-up display will show the details.

5.2. Track Clustering by Audio Features: GrouseFlocks or Bubbletree

For the track clustering task we consider experimenting with two visualizations: a GrouseFlocks graph [8] visualization which was inspired by a post on the Internet [13] and a Bubbletree visualization inspired by another online post []. First step is to partition the tracks into cluster according to their audio features. For the GrouseFlocks visualization we will use agglomerative (hierarchical) clustering. After applying the clustering algorithm we can construct the GrouseFlocks graph. Each leaf node on the graph represents a track. Tracks within a cluster are connected to a node which can represents the genre distributions within that cluster. The colour channel will encode the unique genres which are categorical data. The connecting node will represent a pie chart of the genres distribution. The border outline of the connecting node will be filled with the colour representing the predominant genre for that cluster. The user will be able to click on the nodes that connect the tracks within a cluster and inspect further which tracks are part of that cluster and what genres and audio features are there.

For the Bubbletree (less complicated visualization) we will use some sort of dimensionality reduction algorithm to reduce the audio feature space. We will experiment with the most famous ones like tSNE, PCA or UMAP. Each data point on the graph represnets a track. Each cluster will contain similar tracks and the colour channel will encode the different clusters (all tracks within the same cluster will be coloured with the same colour). Since we want to distinguish different clusters we will use qualitative colour scale.

5.3. Artist Streams Time-Series: Streamgraph

For the task of visualizing the artist streams time-series, we imagine using the streamgraph idiom [8]. The overarching concepts of a streamgraph are relatively simple, but there are many details such as color scheme, ordering, labelling, scale and angle that must be taken into consideration [14]. We are also considering adding some kind of interactivity to the graph. For example, upon selecting an artist in the streamgraph, the streamgraph could morph into another streamgraph showing streams of the tracks of the selected artist through the user's history.

5.4. Top Artists Over Time: Interactive Bar Chart

For this task we are considering an interactive bar chart. This bar chart will display the top 20 streamed artists over the period of their entire history and for specific days so a total of 20 bars. The user can explore their most popular artists for particular days by selecting days in the calendar heatmamp (see Section 5.5). This action will highlight the cell for the corresponding day in the calendar heatmap and will then change the barchart to represent the most popular artists for that day.

The barchart visualization is pretty standard. We will use a horizontal barchart to fit the alignment of the rest of the visualizations. We will use the position on common scale channel to display the data. In addition, we will use the hue channel to highlight specific artist on the barchart when the user hovers over the bars. Current progress of the visualization implementation is shown in Figure 2.

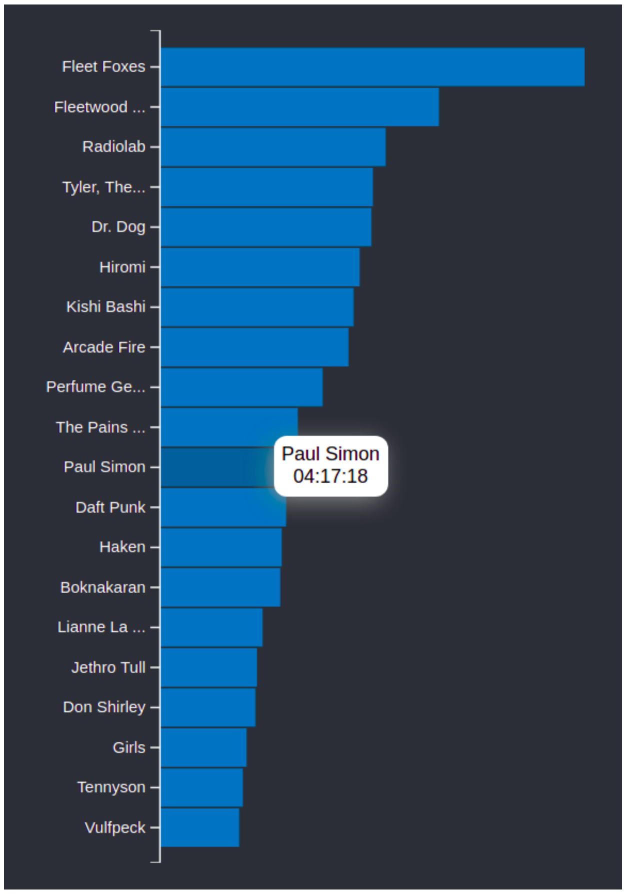

Figure 2. Barchart for top 20 artist

5.5. Daily Listening Pattern: Calendat Heatmap Matrix

For this task a heatmap matrix was chosen to display the listening patterns over time. The visualization is relatively simple and was inspired by the famous visualization for the number of commits per day by Github. There are 7 rows in total and each of them corresponds to the day of the week. Each column of the matrix corresponds to the week number (so a total of 52 weeks) but this will not be displayed.Instead, the year and the month will be displayed along the x axis as they are more descriptive. Since the datasets contain data for each day over a year this visualization will result in a total of 365 cells in the calendar heatmap correspnding to each day of the year. Each cell value corresponds to the listening times in hours, minutes and seconds on the particular day. We use the colour and hue channels to encode the listening times on a particular day. The higher the hue, the higher the interaction time on that day. Since the values of each cell are quantitative sequential data with positive range, a single colour sequential color map will be used.

By aligning the streaming interaction in this way we can clearly observe whether there are any patterns over a month, week or specific days. The user will be able to hover over the cells and a pop up will display the exact listening times for that day. In addition, the user will be able to select particular days of interest which will also highlight the corresponding data for that day in the rest of the visualizations. Current progress of the implementation shown in Figure 3

6. Execution Timeline Milestones

The following timetable suggests a rough estimate with milestones that aims to achieve the proposed objectives in a total of 160 hours.

1. Familiarize ourselves with the dataset - 5 hours each (10 hours total)
2. Familiarize ourselves with D3 - 7 hours each (14 hours total)
3. **Milestone 1**: Create Daily Listening Pattern Heatmap - Inna 5 hours
4. **Milestone 1**: Create Artists Stream-graph - Jonatan 7 hours
5. **Milestone 2**: Add interactivity for Daily Listening Pattern Heatmap - Inna 5 hours
6. **Milestone 2**: Add interactivity for Artists Stream-graph - Jonatan 5 hours
7. **Milestone 3**: Create Artist-Genre Network Graph - Inna Jonatan 8 hours each (16h total)
8. **Milestone 4**: Add interactivity for Artist-Genre Network Graph - Inna Jonatan 5 hours each (10h total)
9. **Milestone 5**: Create Top Artist Over Time Barchart - Jonatan 7 hours
10. **Milestone 5**: Create Track Clustering by Audio Feature - Inna 6 hours
11. **Milestone 6**: Add interactivity for Top Artist Over Time Barchart - Jonatan 7 hours
12. **Milestone 6**: Add interactivity for Track Clustering by Audio Feature - Inna 6 hours
13. **Milestone 7**: Combine all visualizations together - Inna Jonatan 10 hours each (20 total)
14. Final report - Inna Jonatan (10 hours)
15. Presentation and demo - 5 hours

Total of 133 hours. Left 27 hours to spare if something goes wrong.

7. Results

A rough sketch of the Explorify platform shown in Figure 4.

8. Discussion

8.1. Limitations and Future Work

The Explorify platform could be extended to music streaming histories from other platforms such as YouTube or Last.fm. In addition, the Explorify platform could adopt datasets from two or more users and display their similarities and differences. Several people we have talked with have expressed interest in such a feature. In his work on Last.fm explorer, M. Pretzlav shows how this could be done for the data of two users [3]

Our results can be no more accurate or detailed than the data we receive from Spotify. For example, genres only being registered to artists and not tracks hurts how granular we can be in any analysis based on genre. In future work, it could be interesting to use external data source such as MusicBrainz or LastFM to find more granular genre data and see if more interesting results could be found.

References

[1] M. D. M. C. José Bateira, Fabian Gouyon, “Music discovery in spotify with rama,” 2014. 1

[2] T. Dang, A. Anand, and L. Wilkinson, “Fmfinder: Search and filter your favorite songs,” vol. 7431, pp. 348–358, 07 2012. 1

[3] M. A. Pretzlav, “Last.fm explorer: An interactive visualization of hierarchical time-series data.pdf,” 2008. 1, 6

[4] R. Dias, M. J. Fonseca, and D. Gonçalves, “Interactive exploration of music listening histories,” AVI ’12, (New York, NY, USA), p. 415–422, Association for Computing Machinery, 2012. 1

[5] D. Baur and A. Butz, “Pulling strings from a tangle: Visualizing a personal music listening history,” IUI ’09, (New York, NY, USA), p. 439–444, Association for Computing Machinery, 2009. 1

[6] D. Baur, F. Seiffert, M. Sedlmair, and S. Boring, “The streams of our lives: Visualizing listening histories in context,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1119–1128, 2010. 1

[7] J. Wirfs-Brock, S. Mennicken, and J. Thom-Santelli, “Giving voice to silent data: Designing with personal music listening history,” *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020. 2

[8] T. Munzner, *Visualizaiton Analysis and Design*. CRC Press, Taylor Francis Group, 2014. 2, 5

[9] M. Bostock, “D3.js - data-driven documents,” 2012. 4

[10] “React.” `https://reactjs.org/`. Accessed: 2021-10-20. 4

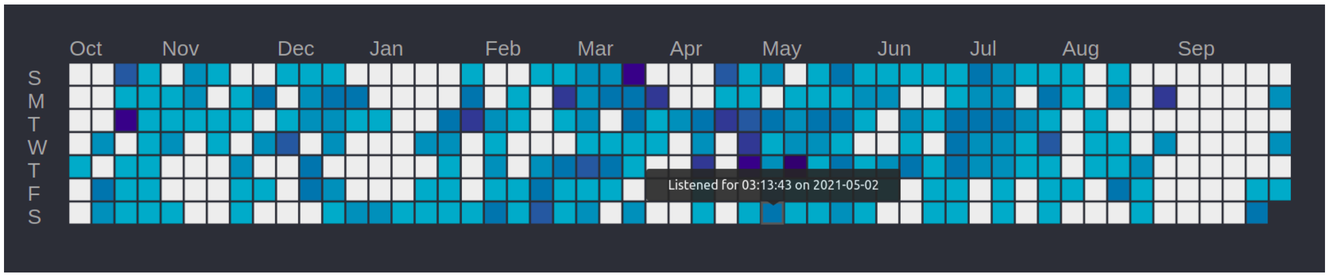

Figure 3. Heatmap for the daily listening pattern of the user. Task specifications described in Section 4.5

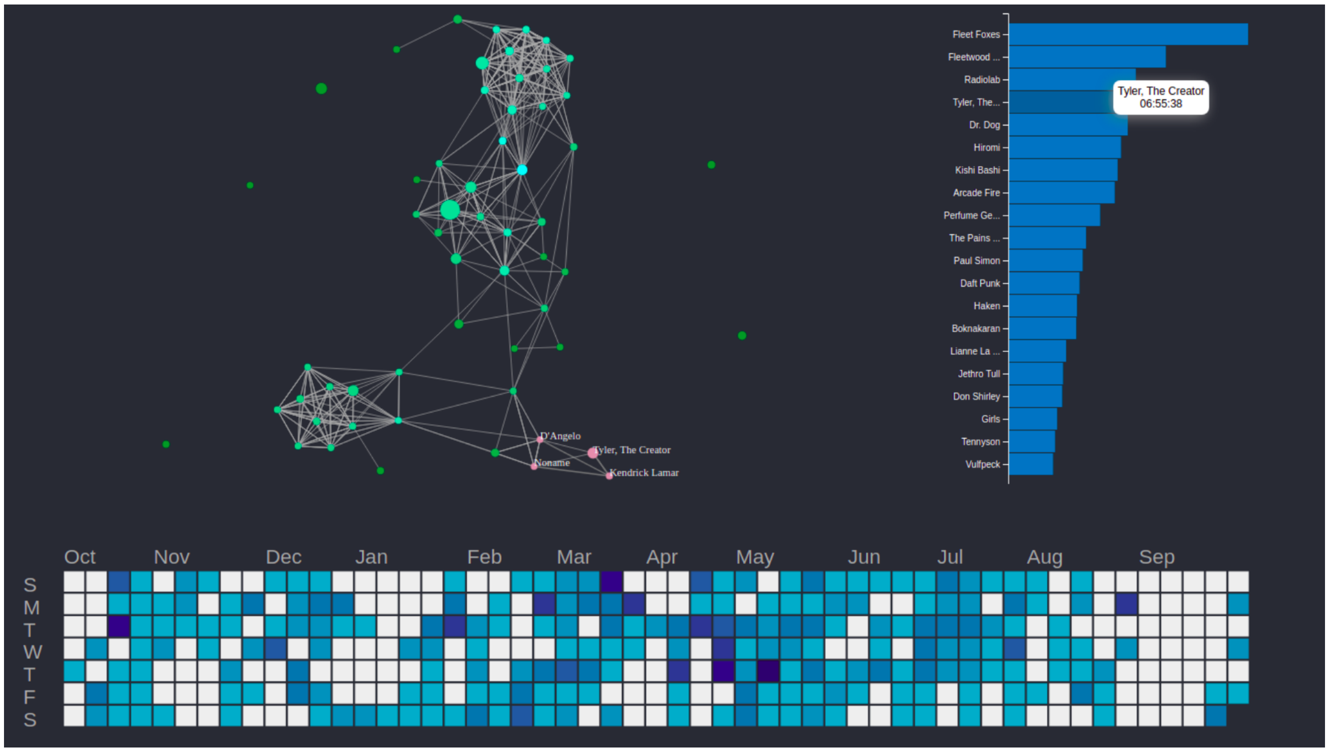

Figure 4. Explorify platform. Current progress of the visualizations.

[11] N. J. Bryan and G. Wang, "Musical influence network analysis and rank of sample-based music.," in *ISMIR*, pp. 329–334, 2011. 4

[12] M. Doerk, S. Carpendale, and C. Williamson, "Edgemaps: Visualizing explicit and implicit relations," 04 2011. 4

[13] "Spotify music clustering." `https://observablehq.com/@sandravizmad/force-directed-layout`. Accessed: 2021-10-20. 5

[14] L. Byron and M. Wattenberg, "Stacked graphs – geometry amp; aesthetics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1245–1252, 2008. 5