FmFinder: Search and Filter Your Favorite Songs

Tuan Nhon Dang, Anushka Anand, and Leland Wilkinson

University of Illinois at Chicago

Abstract. Choices in music express our taste and personality. Different people have different collections of favorite songs. The explosive growth of digital media makes it easier to access any songs we want. Consequently, finding the songs best fit to our tastes becomes more challenging. Existing solutions record user patterns of listening to music, then make recommendation lists for users. By applying information visualization techniques to this problem, we are able to provide users with a novel way to explore their list of recommendations. Based on that knowledge, users can filter the songs according to their needs and compare the music tastes of different groups of people.

1 Introduction

The growth in both quality and quantity of music-sharing websites makes online entertainment increasingly popular. Mining this huge music collection and making selective recommendations for users is a daunting task. To explore this need, some music websites provide an API to allow read/write access to the full slate of music data resources - albums, artists, playlists, events, users, and more. Last.fm API and Amazon.com's API are the best-known of these sites. These APIs give users the ability to build programs to visualize updated music data.

This paper introduces FmFinder, an interactive web-based application that integrates three visualizations using the Last.fm API to enable linking, brushing and filtering. We use a *Tag Cloud* to present the weekly top 50 tracks, which can be use as input for the main visualization. After selecting a track, we use a *Graph Layout* to present recommended tracks. Finally, we can refine the Graph Layout by filtering tracks on the *Dot Plots* [1], *Bubble Plot*, and *Venn Diagram* [2].

The focus of FmFinder is tracks, not artists. Most likely, people love an artist because of his/her songs, but not the reverse. Moreover, when we like an artist, it does not mean we like all his/her songs. It can also happen that we don't like an artist, but there is a particular song of that artist that we really like. Making recommendations based on artists fails in this case.

The main contributions of this paper:

– Most applications recommend songs similar to a selected song by recoding user's listening habits. In this paper, we propose a song recommending algorithm combining user's listening habits (from the LastFm API) and individual preferences.

G. Bebis et al. (Eds.): ISVC 2012, Part I, LNCS 7431, pp. 348–358, 2012.

– We design an interactive visual interface to drill through the recommendation list to find a subset of tracks that better fit user preferences or to compare favorite song collections of different groups of people (different genders, different ages, and different countries).

The rest of this paper is organized as follows. Section 2 provides a brief overview of prior related work. Section 3 describes components of FmFinder and examples. Section 4 concludes the paper.

2 Related Works

Since 2003, the website Last.fm has been directly recording what songs people listen to. Based on these data, Last.fm created a publicly accessible API. Hundreds of Mashups using the Last.fm API have been created on their website for casual users to view. In this section, we provide a brief overview of prior related work using the Last.fm API.

2.1 Visualizing Listening Histories

In 2006, Lee Byron created a minor sensation in the Last.fm and design communities with his stream graph visualization [3] of his own Last.fm listening history. The visualization shows the changing trends in listening history. However, it is very difficult to view details, such as picking a set of artists out of the huge number of artists in a large fixed stream graph.

Last.fm Explorer [4] resolves this issue through interaction. By interacting with the graph directly, users can view different levels of hierarchy, filter the display by date or text searches, view exact data for particular time points, and rearrange the stacked graph to view changes in a particular tag, artist, or track.

In 2010, Dominikus Baur published LastHistory [5], an interactive visualization for displaying music listening histories, along with contextual information from personal photos and calendar entries. It is aimed toward non-expert users and helps them analyze, reminisce and build stories.

Ya-Xi Chen developed this idea further by combining listening histories of multiple users. The paper [6] presents two interactive visualizations which give users a deeper insight into consent and dissent in their listening behaviors, and help them to compare their musical tastes. HisFlocks shows overlaps in genre and artist in certain time periods and LoomFM illustrates sequential listening patterns.

Recently, Martin Dittus, a former Last.fm employee, grabbed listening data for staff, moderators, and alumni, and visualized 8.7 million scribbles in an calendar heat map [7]. It is like a calendar view, but instead of days, the interest is centered around hours of the day.

2.2 Recommending Artists

In 2005, Frederic Vavrille created Liveplasma [8], a flash-visualization based on Amazon's e-commerce service that explores links between music artists, movies,

directors, and actors. It maps and displays music and movie search results with linkages and groupings. Figure 1 shows an example. After the search term is submitted, it is immediately surrounded by other artists. The graph becomes complicated when there are many similar artists/movies. Liveplasma does not offer a way to filter data in this case.

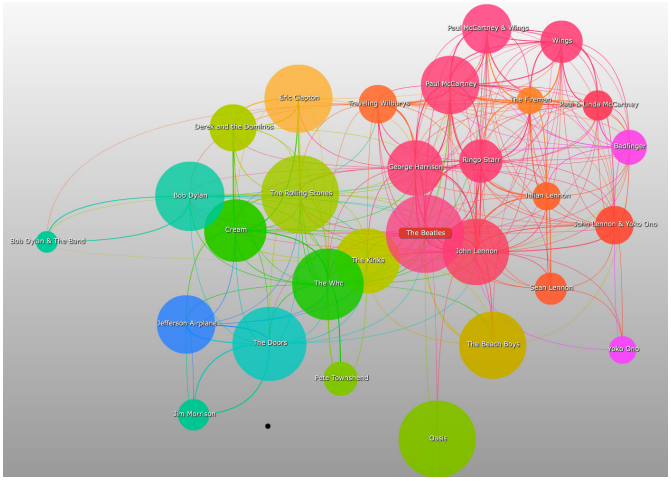

Fig. 1. Liveplasma: Artists related to "The Beatles"

TuneGlue [9] is based on a similar idea. TuneGlue is a force-directed graph exploring relationships between artists based on the Last fm database and amazon.co.uk. One can start by inputting an artist name and then the network is built with only one root node. By clicking the root node, one can expand other related artists. One can also explore all albums from each artist on the spot and link out to amazon.com to buy any of them. However, this function does not seem to be working at the moment.

2.3 Recommending Songs

The Genius Sidebar, introduced in iTunes 8, automatically generates a playlist of songs from the user's library which contains songs similar to the selected song. Genius playlists are created by the ratings system and collaborative filtering. An iTunes Store account is required because information about the user's library must first be sent to Apple's database. Algorithms determine which songs to play based on other users' libraries.

Different from the Genius Sidebar, FmFinder presents recommended songs in a graph layout instead of a list view. Moreover, users can filter all similar songs manually on a dot plot, a bubble plot, and/or a venn diagram. No usage history is required. Users can change their preferences at will. The next Section describes components of FmFinder and examples.

3 FmFinder

The FmFinder GUI incorporates three major components as depicted in Figure 2: Input Panel includes Box 1 and Box 2, Output Panel includes Box 3, and Filtering Panel includes Box 4, Box 5, Box 6, and Box 7.

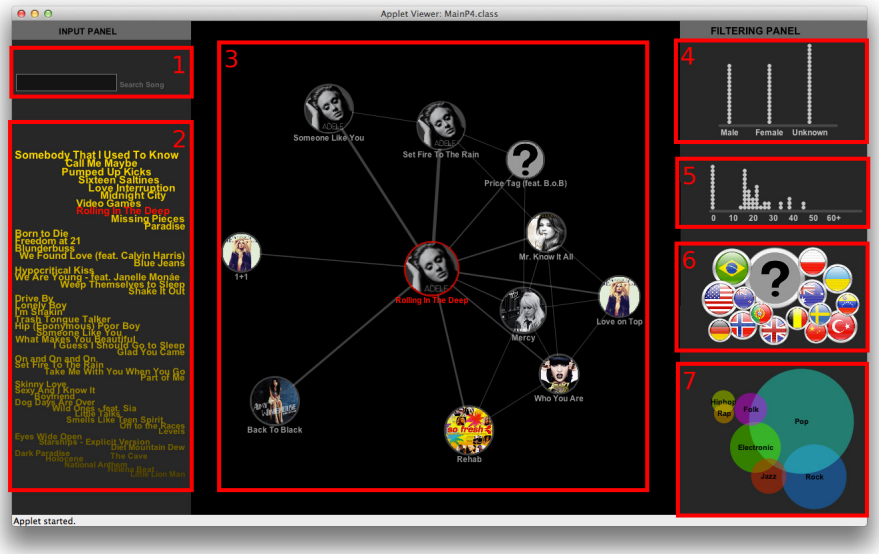

Fig. 2. FmFinder GUI: search box (Box 1), tag cloud (Box 2), graph layout (Box 3), gender dot plot (Box 4), age dot plot (Box 5), country bubble plot (Box 6), and genre venn diagram (Box 7)

Information visualization systems should allow one to perform analysis tasks that largely capture people's activities while employing information visualization tools for understanding data [10]. In the rest of this section, we describe five basic analysis tasks implemented in FmFinder: searching (using the search box), retrieving value (retrieving track information by brushing a node in the graph layout), sorting (sorting tracks by the level of recommendation), characterizing distributions (using Dot Plots, Bubble Plots, and Venn Diagrams to show statistics of the selected song), and filtering (filtering tracks by user preferences).

3.1 Input Panel

The initial layout contains only Input Panel (Output Panel or Filtering Panel are empty). Input Panel enables the user to select a song name. Users can input a song by using the search box (Box 1 in Figure 2) or select a song from the weekly top 50 songs showing in the Tag Cloud (Box 2 in Figure 2). Colors and font sizes in the Tag Cloud are determined by popularity of songs.

3.2 Output Panel

After we have input a song, related songs are displayed in a node-link representation (Box 3 in Figure 2). In particular, we have selected the track "Rolling In The Deep" ranked 8 in the Tag Cloud. The track "Rolling In The Deep" now becomes the root node in the graph (highlighted in red), surrounded by top ten similar tracks. The picture on every node displays the artist of the song labeled right under it. These pictures help in recognizing tracks of the same singer. The sizes of nodes are based on popularity of the track. Additionally, two nodes are linked if they suggest each other. The thickness of the edges indicates the level of recommendation. Edge length is not used to encode any information. The nodes organize themselves to reduce link-crossed.

There are several ways that users can interact with the application. Users can drag nodes to reorganize the graph, use a slider to control the number of similar tracks to be displayed in the graph, brush a node to request detailed information, or play a track. In the example in Figure 3, we have brushed the node "Mr. Know It All". The nodes not directly connected to the brushed node are faded so that we can focus on the relationships of the selected node. Moreover, the brushed track is downloaded and played at the bottom. The details of the brushed track are displayed next to the video. Notably, we can make the brushed track become the root node by a simple click.

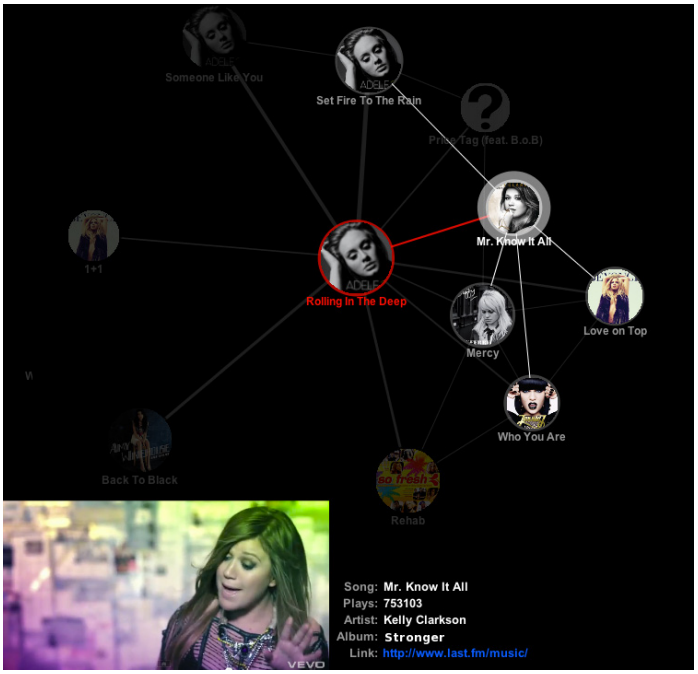

Fig. 3. Node-link brushing: "Mr. Know It All"

3.3 Filtering Panel

Music tastes changes with time, and sometimes depends on one's mood. The songs that I like to listen to today might not be my favorite songs from yesterday. FmFinder allows users to drill through the recommendation list to find a subset of tracks that better fit their current preferences.

Preview the Selected Song. After we have selected a track, the recommended tracks are plotted in the node-link diagram. The statistics of the selected track are also plotted in the Filtering Panel. We use a set of interactive visualization methods to present statistics of the selected track: Dot Plots, Bubble Plot, and Venn Diagram.

Figure 4 shows an example. We have selected the leading track of current week (the first track in the Tag Cloud),"Somebody That I Used To Know". The Dot Plot in Figure 4(a) shows the distribution of the top 50 listeners by gender. Notice that most listeners are female. The Dot Plot in Figure 4(b) shows the density distribution of the top 50 listeners by age. Some listeners not wanting to specify their ages are given the default value 0. Moreover, listeners to this track are divided into 2 groups, from 16 to 24 and from 30 to 36 year olds.

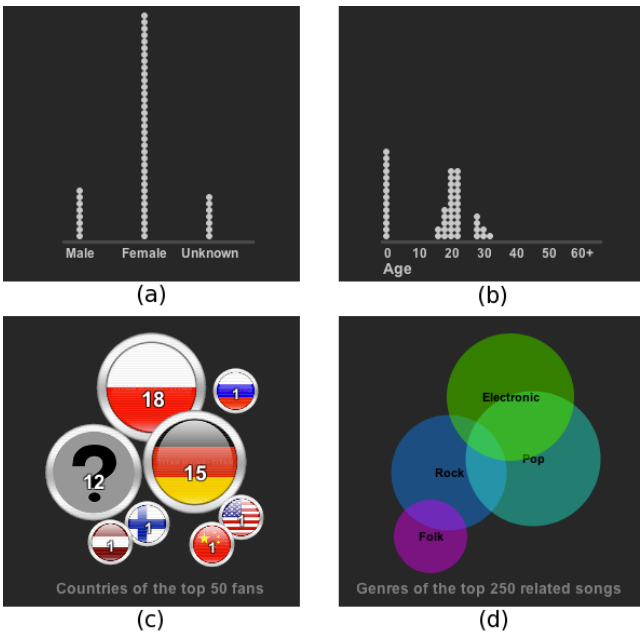

Fig. 4. Filtering Panel, statistics of "Somebody That I Used To Know": (a) Gender Dot Plot of the top 50 fans (b) Age Dot Plot of the top 50 fans (c) Country Bubble Plot of the top 50 fans (d) Genre Venn Diagram of the top 250 related songs

A single dot in the Dot Plots represents a single listener. Dot plots, instead of Histograms or Bar Charts, are used to present statistics of the top 50 listeners because they allow to select every single listener to check out his/her compete profile, including user id, name, age, gender, and nationality. In case there are too many listeners in the result to be displayed as single dots, we can make the dots overlapped vertically to fit the allowed space. This makes a Dot Plot look very similar to a Histogram. However, brushing every single dot is still possible.

Country data is represented in a Bubble Plot which emphasizes their visual appearance. Figure 4(c) shows the distribution of the top 50 listeners by country. Each bubble presents one country. Country flags embedded on top of each bubble facilitate identification. The bubble sizes are decided by the number of listeners from that country. One can request to show the number of listeners on the top of each flag as shown in Figure 4(c). As we notice, there are 12 listeners with non-specified nationality, 18 listeners are from Poland, 15 listeners are from Germany, and the rest are from Russia, Latvia, Finland, US, and China. Overall, 36 over 38 known-nationality listeners are from Europe. Gotye, the singer of "Somebody That I Used To Know," is a Belgian-Australian musician and singer-songwriter, so it is not surprising why the song is popular in Europe.

Figure 4(a), Figure 4(b), and Figure 4(c) are linked. By selecting a country from the Bubble Plot in Figure 4(c), all listeners from that country are highlighted in Figure 4(a) and Figure 4(b).

There are top 250 songs related to the selected song, "Somebody That I Used To Know", returned from LastFm API. We use a Venn Diagram to present the genre distribution of the top 250 related songs. As depicted in Figure 4(d), most of related songs are pop and/or electronic, rock. The genres of a song are retrieved by song tags by listeners. Many songs are mixtures of different genres. Therefore, a Venn Diagram is an appropriate representation for such intersections.

Filter Related Songs. Previewing the selected song provides a guideline to filter related songs. We filter and rank songs based on the following features:

- Number of fans who pass filtering conditions (Let F be the number of fans, F receives a value from 0 to 50. LastFm returns at most 50 top listeners of a selected song).
- Play counts of the songs (Let P be the play counts, P receives a value from 1 to several millions).
- Similarity ranking from Last.fm (Let S be the similarity ranking from Last.fm, S receives a value from 1 to 250. S equals 1 when the song is most similar to the selected song).
- Genres of the songs. These genres are obtained by parsing song tags by listeners.

The weighting function is computed by the following equation:

$$Weight(song) = F * \sqrt[10]{P} - \frac{S}{5} \tag{1}$$

In Equation 1, we favor the popular songs and LastFm rankings. However, F is the most important factor to decide weight of a song. In particular, the $\sqrt[10]{P}$ mostly produces the values from 3 to 5. For the same S, the least popular song with f fans who pass filtering conditions still get a higher weight than the most popular song with half of f. This is how FmFinder can be used to discover new related songs.

Because LastFm returns at most 250 similar songs ($S \leq 250$), $\frac{S}{5}$ produces the values from 1 to 50. Subtraction from this amount allows more similar songs to get higher weights. Finally, we order the weights to obtain a ranking of the related songs.

More importantly, genres of songs need to be in the set of active genres (selected on the Venn Diagram). Otherwise, $Weight(song)$ is reset 0.

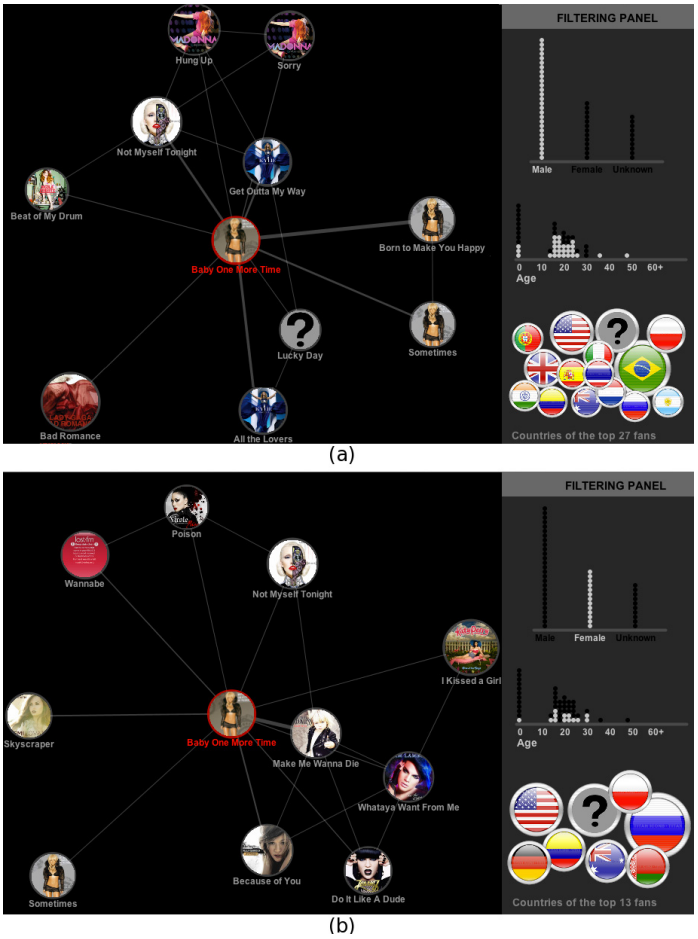

Fig. 5. Filtering recommended songs by gender: a) Male b) Female

Figure 5 shows a filtering example. We have selected "Baby One More Time" of Britney Spears as the root node. The Dot Plots and Bubble Plot now show statistics of the selected song "Baby One More Time". We then apply the filter on gender. Figure 5(a) shows recommended songs for males. Figure 5(b) shows recommended songs for females. The results are different recommended lists. Males like to listen to Madonna, Lady Gaga, and Kylie Minogue while females prefer The Pretty Reckless, Pink, and Kelly Clarkson. Notice that the Dot Plots and Bubble Plot are also updated based on filtering conditions.

Figure 6 shows the distribution of the top 50 listeners by gender of two songs preferred by males and two songs preferred by females regarding the selected song "Baby One More Time" of Britney Spears. The charts help to understand how FmFinder makes decisions resulting in Figure 5.

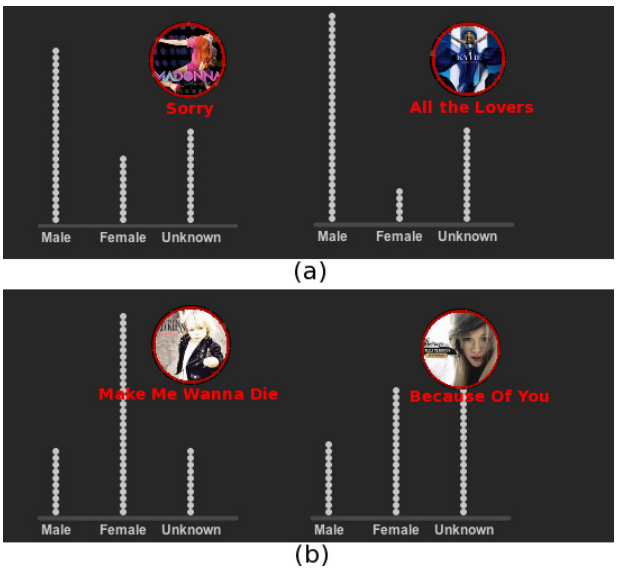

Fig. 6. Distribution of listeners by gender: (a) Songs preferred by males (b) Songs preferred by females

Figure 7 shows another filtering example. We have selected "Baby One More Time" of Britney Spears as the root node. We then apply the filter on genre. The filtering condition is set up on the Venn Diagram (active genres are highlighted, inactive genres are faded). Figure 7(a) shows recommended Rock, Rap, and Hiphop songs . Figure 7(b) shows recommended Pop and Electronic songs. The results are completely different recommended lists.

3.4 Conclusions

FmFinder is an web-based application built in Processing to address the challenges of searching music that fits user preferences. Some distinct features of FmFinder compared to prior related works are:

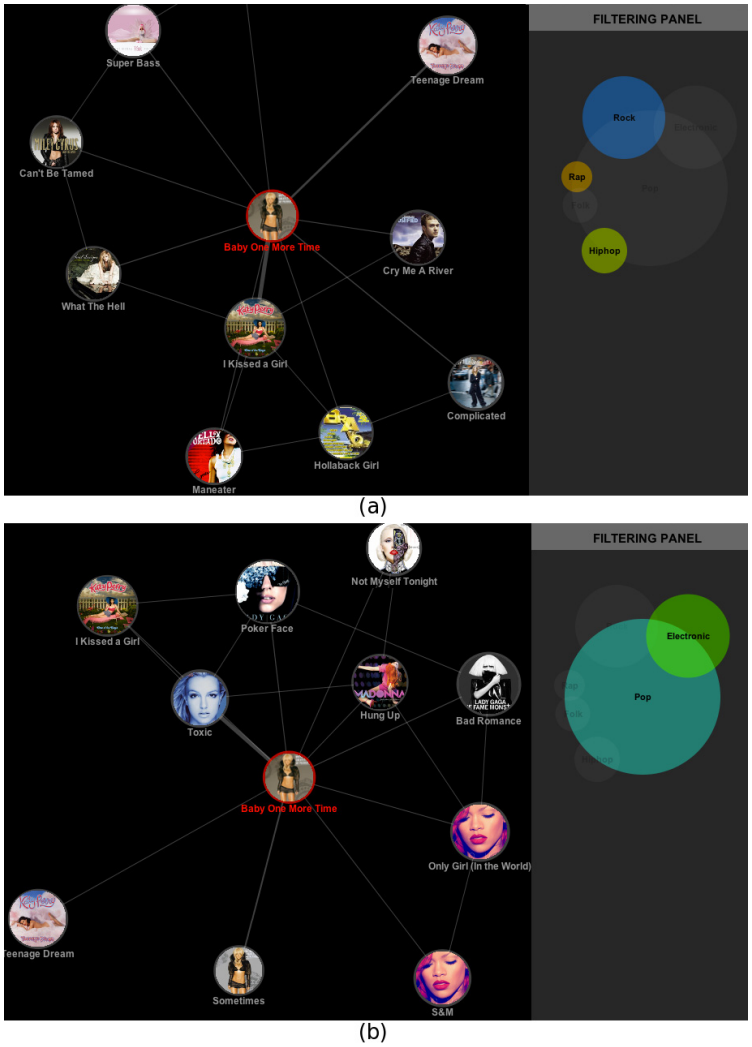

Fig. 7. Filtering recommended songs by genre: a) Rock, Rap, and Hiphop b) Pop and Electronic

- FmFinder works directly on song recommendations, not artist recommendations.
- FmFinder combines multiple visualization techniques in all three processes (Input, Filtering, and Output).
- Users can set their own preferences manually or filter and compare favorite songs of different groups of people.

FmFinder allow users to perform basic analysis tasks required by any information visualization tools:

- Searching an online database (LastFm) to find a song that users require.
- Retrieving track information and/or playing a track by brushing a node in the graph layout.
- Sorting tracks by the level of recommendation which is then encoded by the thickness of edges in the node-link diagrams.
- Characterizing distributions of the top listeners by gender, age, country using Dot Plots, Bubble Plot, and distributions of the related tracks using Venn Diagram.
- Filtering tracks by user preferences.

References

1. Dang, T.N., Wilkinson, L., Anand, A.: Stacking graphic elements to avoid overplotting. IEEE Transactions on Visualization and Computer Graphics 16, 1044–1052 (2010)
2. Wilkinson, L.: Exact and approximate area-proportional circular Venn and Euler diagrams. IEEE Transactions on Visualization and Computer Graphics (2011) (in press)
3. Byron, L., Wattenberg, M.: Stacked graphs - geometry & aesthetics. IEEE Transactions on Visualization and Computer Graphics 14, 1245–1252 (2008)
4. Pretzlav, M.A.: Last.fm explorer: An interactive visualization of hierarchical time-series data. Tracks A Journal of Artists Writings 7 (2008)
5. Baur, D., Seiffert, F., Sedlmair, M., Boring, S.: The streams of our lives: visualizing listening histories in context. IEEE Transactions on Visualization and Computer Graphics 16, 1119–1128 (2010)
6. Chen, Y.X., Baur, D., Butz, A.: Gaining musical insights: Visualizing multiple listening histories. In: Workshop on Visual Interfaces to the Social and Semantic Web, VISSW 2010 (2010)
7. Dittus, M.: Revealing the periodic listening habits of last.fm users (2011)
8. Vavrille, F.: Liveplasma: Quickly discover similar movies and songs (2005)
9. Team, O.: Music plasma: Music relationship explorer (2006)
10. Amar, R., Eagan, J., Stasko, J.: Low-level components of analytic activity in information visualization. In: Proc. of the IEEE Symposium on Information Visualization, pp. 15–24 (2005)