Pulling Strings from a Tangle: Visualizing a Personal Music Listening History

Dominikus Baur
Media Informatics, University of Munich
Munich, Germany
dominikus.baur@ifi.lmu.de

Andreas Butz
Media Informatics, University of Munich
Munich, Germany
andreas.butz@ifi.lmu.de

ABSTRACT

The history of songs, to which a person has listened, is a very personal piece of information. It is a rich data set that comes as a byproduct of the use of digital music players and can be obtained without interfering with the user.

In this paper, we present three visualizations for this data set and a mechanism for generating new playlists from the user's own listening history, based on a navigation metaphor. First, temporal proximity is interpreted as a simple similarity measure to lay out the entire history on a two-dimensional plane. Closed listening sessions are then used to make chronological relations visible.

The generated playlists mimic the user's previous listening behavior, and the visualizations make the automatic choices understandable, as they share visual properties with the history. In this sense, our visualizations provide a visual vocabulary for listening behaviors and bring scrutability to automatic playlist generation.

Author Keywords
Listening history, visualization, playlist creation, navigation metaphor

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION

The size of personal digital music collections is constantly increasing and a wide variety of affordable portable music players allows listening to music wherever we go. On top of this, online services now also provide free music streaming, creating an abundance of available music. As users are increasingly overwhelmed by this situation, companies as well as researchers are investigating ways to handle it properly. They mainly focus on tools for organizing and visualizing large music collections.

Two general directions have been explored in this context. Categories, such as artist, album or genre provided by the user (mostly in the way of ID3-tags) are used to categorize music in a hierarchical fashion. Most software media players use this approach and provide alphabetized lists of songs, that can be sorted and filtered based on these categories. The second approach uses machine-learning to automatically find similarities between songs based on their actual audible content. Low-level feature analysis and sophisticated weighting schemes provide a similarity metric, that can be used, e.g., to create a self-organizing map [8]. The promise of this approach is to display songs, which sound similar, in close spatial proximity.

Both approaches have their downsides. Categories let the user easily retrieve songs, for which they remember artist, album or title. Finding similar songs, e.g., to create a coherent playlist, is much more difficult and not well supported by these interfaces. Instead, the user has to possess a deep knowledge about his own collection to manage this task. A feature-based approach naturally alleviates these problems, but creates new ones. The algorithms in this field are not yet perfect and may never be [2]. Moreover, similarity is mostly based on predefined feature sets and weights, which do not necessarily reflect the user's ideas about similar songs.

PERSONALIZATION

Personalization is a central issue for improving the accuracy of music recommendation systems. While most of them use either audio features or even manual analysis (Pandora Internet Radio[1]), a new group of recommenders is based on the so-called "crowdsourcing", relying on a community to produce similarity values. In a first step, preferences of users are identified, e.g., by tracking shopping habits (Amazon[2], Apple iTunes[3]) or listening history (Last.fm[4]). Then, missing intersections between pairs of users with very similar histories are computed and turned into recommendations. Listening histories have been used for creating personalized playlists in several research projects [1, 4, 10], and more recently in the "Genius" feature in iTunes[5].

[1]www.pandora.com
[2]www.amazon.com
[3]www.itunes.com
[4]www.last.fm
[5]www.apple.com/itunes/features/#genius

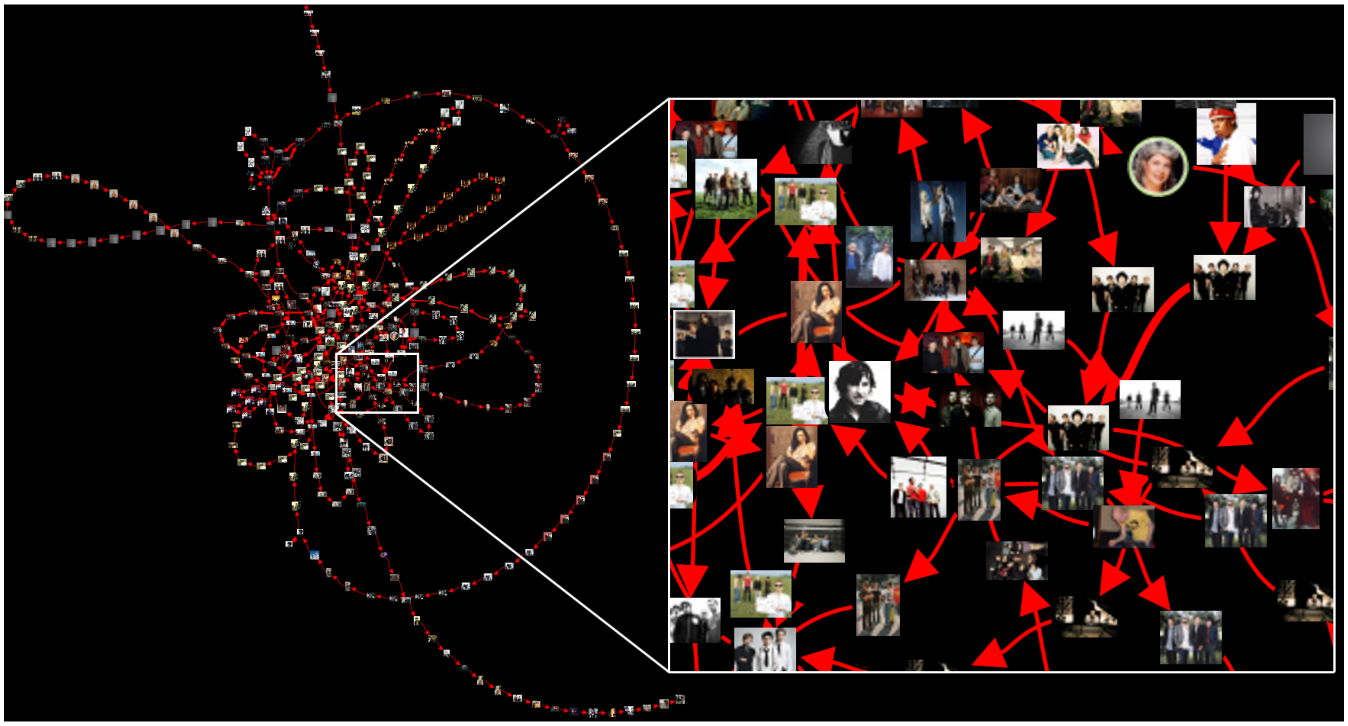

Figure 1. The neighborhood in a listening history serves as a similarity metric and forms the basis for the *Tangle* visualization.

Lambiotte et al. [7] explored listening histories to extract groups of music and also built a visualization of the underlying networks, but focused on the group aspect. Lee Byron[6] and LastGraph[7] visualize Last.fm listening histories using so-called streamgraphs. These graphs contain information about artist composition for time periods thus providing only a very high-level and non-interactive overview of listening habits. To our knowledge, listening history on a song level has never been used for the interactive visualization of an entire personal music collection, especially taking into account the temporal aspect. We present three visualizations for this type of data that let a person not only analyze her listening behavior, but also discover patterns, compare behaviors and create new playlists.

LISTENING HABITS

The listening history describes the past usage of a music player by a person. It is a readily available type of data that is created as a byproduct of just listening, but provides a very personal and important view. If we observe user behavior on a very low level, only four types of interaction are possible: 1) Directly choosing a song or a prepared playlist to start playing, 2) Letting the system decide about the next song (either orderly or randomly), 3) Skipping the current song to hear the following one, 4) Stopping the listening session by using a stop-button or exiting the software.

A few important usage patterns can be observed from these simple interactions: A listening history can normally be divided into listening sessions of differing lengths by either setting a fixed time threshold to separate sessions or using a more sophisticated approach similar to what Cooper et al.[5] used for finding "events" in a photo collection. If the system randomly chose a song that the user does not like, she might fall into repeatedly pressing the Next-button until a more enjoyable song comes up (a behavior that can also be used for creating playlists[9]).

This means that every listening session of the user can be described as a playlist: It is manually created, either beforehand or on-the-fly, either by directly choosing a preferred song for the current situation, or by skipping the system's suggestions until a fitting one is reached. Only songs that were played as a whole are logged. Songs that follow one another in this adjusted history therefore can be assumed to have some connection in the user's mind, even if they are only the lesser evil. "Satisficing", i.e., stopping a search once a suitable candidate comes up and not continuing to find the best available solution is a fitting description and can be observed in music as well as image collections[3].

VISUALIZING THE LISTENING HISTORY

A system that provides us readily with this kind of filtered data is Audioscrobbler[8], the tracking software of Last.fm. It works with a demon process and plug-ins for common media players, that log listening behavior and transmit it to the Last.fm server, to make it accessible via web services and

[6]www.leebyron.com/what/lastfm
[7]lastgraph3.aeracode.org
[8]www.audioscrobbler.net

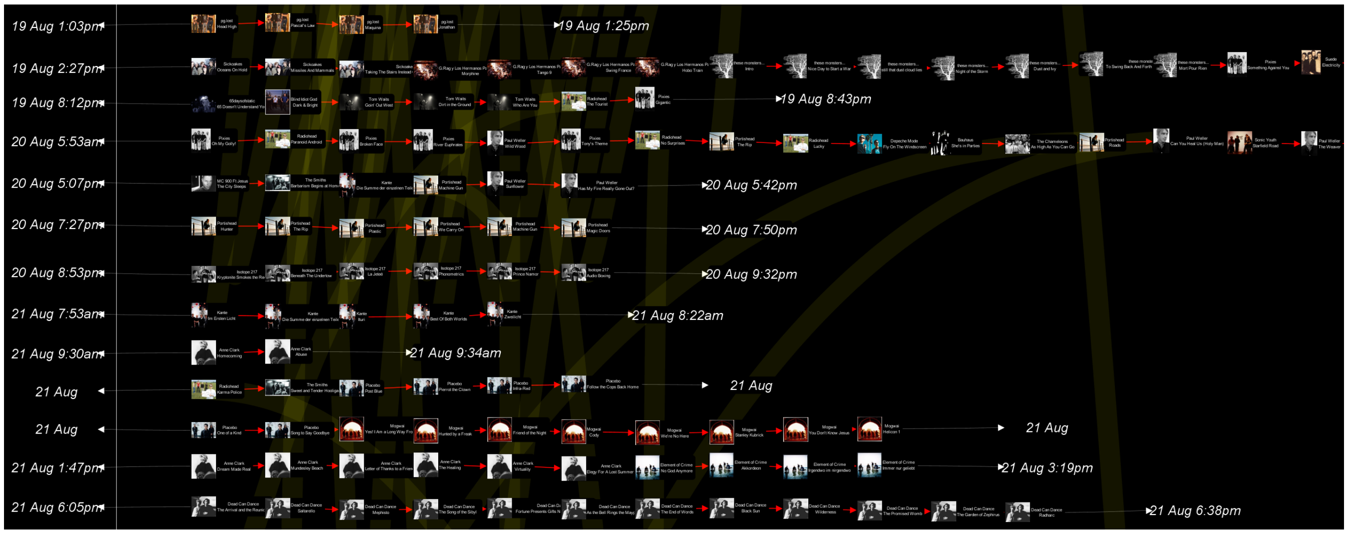

Figure 2. The separate listening sessions are represented as *Strings*.

their home page. Fittingly, only songs that were listened to in their entirety are recorded. This data is sorted chronologically and provides to the minute timestamps for all songs that were heard during the current year (date only for older songs). Meta data, such as artist, album title, year of release etc. can be easily retrieved from Last.fm.

As this kind of data is very complex and extensive (a regular listener might play thousands of songs during one year), we use Information Visualization techniques to cope with it. In the following, we present three visualizations that allow a user to discover different patterns in this data.

Global overview: *Tangle*

A first visualization, called *Tangle*, provides a global view of all songs in one listening history. We make the assumption, that a person has listened to all songs in her collection at least once (so every song is visible), which makes this visualization a tool to work with a complete music collection. The listening history is represented by one long sequence of songs, as people normally listen to only one song at a time (at least voluntarily). By itself, this sequence is only marginally interesting. Finding connections between songs (e.g., identical ones, songs by the same artist etc.) can provide much richer information. As described above, a listening history can serve as a similarity measure between songs. If two songs were listened to at least once after one another, they are assumed to have some relation.

Tangle (see figure 1) represents the entire music collection as a node-link-diagram, in which every node is one individual song and every edge a direct neighborhood between two songs in the history. As the layout is produced in a force-directed way, a pseudo-physics mechanism that leads to rejection between unconnected nodes and attraction between nodes sharing an edge, spatial proximity between two nodes reflects (roughly) their perceived similarity. This type of visualization looks at first sight confusing, but provides even in a non-interactive form several kinds of information: Songs are represented by photos of their performing artists, so a quick visual subdivision becomes possible. Songs and sequences that were listened to in isolation group themselves as loops at the border. The visual thickness of the edges encodes how often this combination of songs was listened to and is visible at a glance. The direct neighborhood of a song contains its most similar neighbors (at least according to the user's implicitly expressed taste).

To learn the details of the history interaction is necessary: The user can pan and zoom the display, get additional information (name of artist and song) via tooltips, drag single nodes to any position and fix them there for closer analysis (the underlying physics simulation lets the neighboring nodes follow) or filter the visualization for one node and two levels of its neighbors by hovering above it.

While this visualization shows the readily available relations between songs, it is not suitable for analyzing temporal patterns in a listening history. Therefore, we created an additional visualization named *Strings* for exactly this purpose.

Session-based view: *Strings*

As explained above, a listening history can be subdivided into listening sessions of varying lengths. In the *Strings* visualization we exploit this fact and display these sessions as horizontal strings of songs that are sorted chronologically from top to bottom (see figure 2). Every song is shown with a picture of its performing artist, making visual grouping easily possible. Labels for different months and years provide orientation even at high zoom levels. In this type of visualization, identical songs are no longer represented by one node, but possibly by several nodes in different Strings. Identical songs are connected by semi-transparent wide yellow lines. In terms of interaction, the user can again pan and

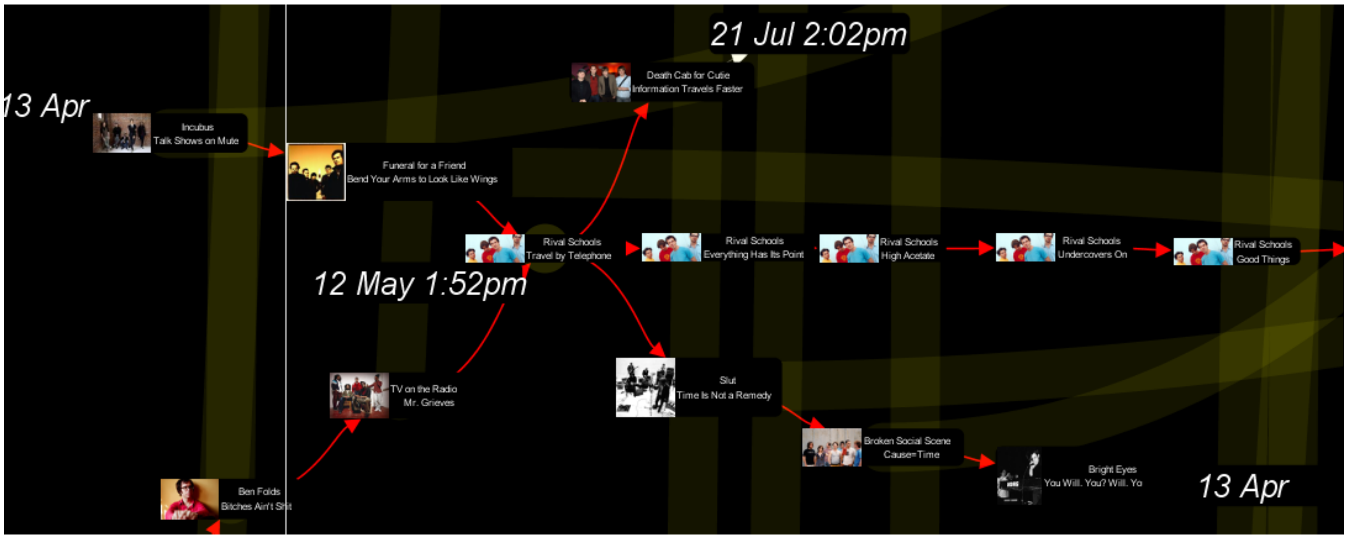

Figure 3. *Knots* focus on one song which potentially appears within several *Strings*.

zoom the whole display to get an overview or focus on one certain section of time. To explore changes in the layout, the user can manually reposition nodes if necessary. A click into empty space returns to the original layout.

Strings show information about listening habits in a temporal context: Patterns of very long or very short sequences, weeks of pauses during which no music was logged (e.g., while being on vacation), and sequences containing only songs by one artist (leading to every song having the same image) are all directly visible. Also, the faint connections between identical songs show phases in which a song was listened to repeatedly in a short period of time or over and over again, as well as continual identical sequences of songs, such as predefined playlists or albums (which tend to be listened to as a whole). Different patterns and types of listeners can be recognized at a glance: People who listen to music during fixed times (e.g., subway rides to work, logging the listening history on their iPods) have regular sequences on every weekday. Special occasions, such as parties, appear as unusually long sessions. An album- or playlist-centric listener rather has vertical connections between the sessions, while a listener who relies on skipping through a randomized list produces more chaotic lines. This provides a visual vocabulary for the high level listening behavior and makes it possible to compare, for example, the listening history of different users in this respect.

Connecting *Strings* by *Knots*

Selecting a node in the *Strings* view provides a fall-back to a variant of the *Tangle* visualization, using one node per song, but only in a local and not a global fashion. Coming from one song, all identical songs in all affected sequences are merged using a force-directed layout (we call this a *Knot*, see figure 3), so repetitions, such as playlists or albums become visible. As the date labels for the start and end of a sequence are attached to the first and last song, they stay recognizable. This mechanism is an alternative to visually following the yellow trails, if they become too long (e.g., for the same song in the first and last sequence) or if there are too many connections. Additionally, patterns, such as short loops of several songs are hard to see if the songs are played in a different order every time, but by clicking on one of them, they become easily recognizable.

Creating playlists

In both visualizations, users can create playlists that will reflect their own taste. Based on a map metaphor, as found in navigation software (e.g., Google Maps[9]), the user can choose a start and an endpoint. The system then automatically creates a playlist between these two by following existing edges between them. Additionally, the user can determine "waypoints", i.e., songs that should also be passed, and the system refines the "route" accordingly. Because a song might have more than one incoming and outgoing edge, that were visited with different frequencies, the user can also adjust the novelty factor of the created playlist. The higher this factor, the more unusual the path between two songs will be.

As the existing edges represent the user's listening behavior, the created playlists also reflect that: Listeners who prefer listening to albums will find snippets of these albums in their playlists, just as user-defined playlists will also partially appear in automatically created ones. This can, for example, preserve the order of songs, which might be important. People with more randomized listening sessions will get playlists with the same tendency. A great advantage of this playlist creation mechanism is its simplicity. No complex algorithms or user models are needed, as the musical taste automatically arises from the user's history without the need for making it explicit. The routing and waypoint metaphor is simple and familiar, so the user can adjust created playlists and has a direct feedback why they contain the

[9]maps.google.com

songs they do. This quality of a system to make its decisions and the reasons for them clear, is often called scrutability[6].

SUMMARY, LIMITATIONS AND FUTURE WORK

We have presented three visualizations for a personal music listening history, one representing song similarity on the basis of temporal proximity as spatial proximity in a two-dimensional layout, and the other two uncovering listening patterns by emphasizing listening sessions. Additionally, we have applied a routing and navigation metaphor to our visualizations to let users create customized playlists in a simple and scrutable way.

While we have used all of these visualizations to explore various user histories successfully within their limits, we are still lacking a user evaluation for the playlist creation. As we argued above, every new playlist is actually a patchwork of existing session snippets and should thus reflect the user's taste as reflected in her listening history. Therefore, it should be easily possible to let the system create several playlists and let the users rate them. We plan to evaluate this in a study as soon as possible.

Our current implementation also lacks true scalability. This is caused by the implementation as well as the visualization concept and interaction techniques. Written in Java using the prefuse framework[10], our prototype can currently display up to 1.000 songs without loosing a tolerable frame rate. Connections between similar songs in *Strings* become untraceable with even less songs and *Tangle* is confusing with a hundred unique items. At the moment the solution is to display only songs from a certain time period making a bulk of short-term information accessible but hiding long-term aspects (e.g., the user rediscovered a song she listened to repeatedly two years ago). We want to address this problem in our future work.

Including additional kinds of data could prove to be very interesting: One fact that we learned while browsing through listening histories is, that while people tend to listen to albums as a whole, they also have a tendency to skip the same songs in every listening session. This is also the case, although much less frequently, for user-created playlists. It would therefore be interesting to include these manually created listings (for albums directly available on the Last.fm page) into the visualization to see corresponding patterns.

Also, in order to pick up the community idea behind Last.fm, it might be interesting to combine listening histories for different users into a single visualization to make them directly comparable. Intersections could be enhanced visually and data from different users be color coded to let users perform the above mentioned recommendation functions (i.e., looking for outliers in otherwise similar histories) themselves. To move more into the direction of Lambiotte's work [7], parallel visualizations of different users could be used to find overall tendencies and groups with similar taste.

[10]prefuse.org

Finally, the visualizations and the playlist creation could be merged with a media player, to expand the now rudimentary playback functions and work as an access mechanism for the whole collection. In this case, songs that are not yet in the collection (because they were never listened to) have to be displayed in a tentative way, maybe based on an existing community-based similarity measure.

ACKNOWLEDGMENTS

This work was funded by the University of Munich and the state of Bavaria. We would like to thank W. Bradford Paley for visual and Clifford Beshers for lexical inspiration.

REFERENCES

1. A. Andric and G. Haus. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools Appl.*, 29(2):127–151, 2006.

2. J.-J. Aucouturier and F. Pachet. Improving timbre similarity: How high's the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

3. F. Bentley, C. Metcalf, and G. Harboe. Personal vs. commercial content: the similarities between consumer use of photos and music. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 667–676, New York, NY, USA, 2006. ACM.

4. H.-C. Chen and A. L. P. Chen. A music recommendation system based on music data grouping and user interests. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 231–238, New York, NY, USA, 2001. ACM.

5. M. Cooper, J. Foote, A. Girgensohn, and L. Wilcox. Temporal event clustering for digital photo collections. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(3):269–288, 2005.

6. J. Kay. Stereotypes, student models and scrutability. In *ITS '00: Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, pages 19–30, London, UK, 2000. Springer-Verlag.

7. R. Lambiotte and M. Ausloos. Uncovering collective listening habits and music genres in bipartite networks. *Physical review*, 72(6), Dec. 2005.

8. E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. Technical report, Vienna University of Technology, 2001.

9. E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behaviour. In *Proc. of the 6th ISMIR Conference*, pages 634–637, 2005.

10. S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *Proceedings of the Third International Conference on Music Information Retrieval. Paris: IRCAM*, pages 222–230, 2002.