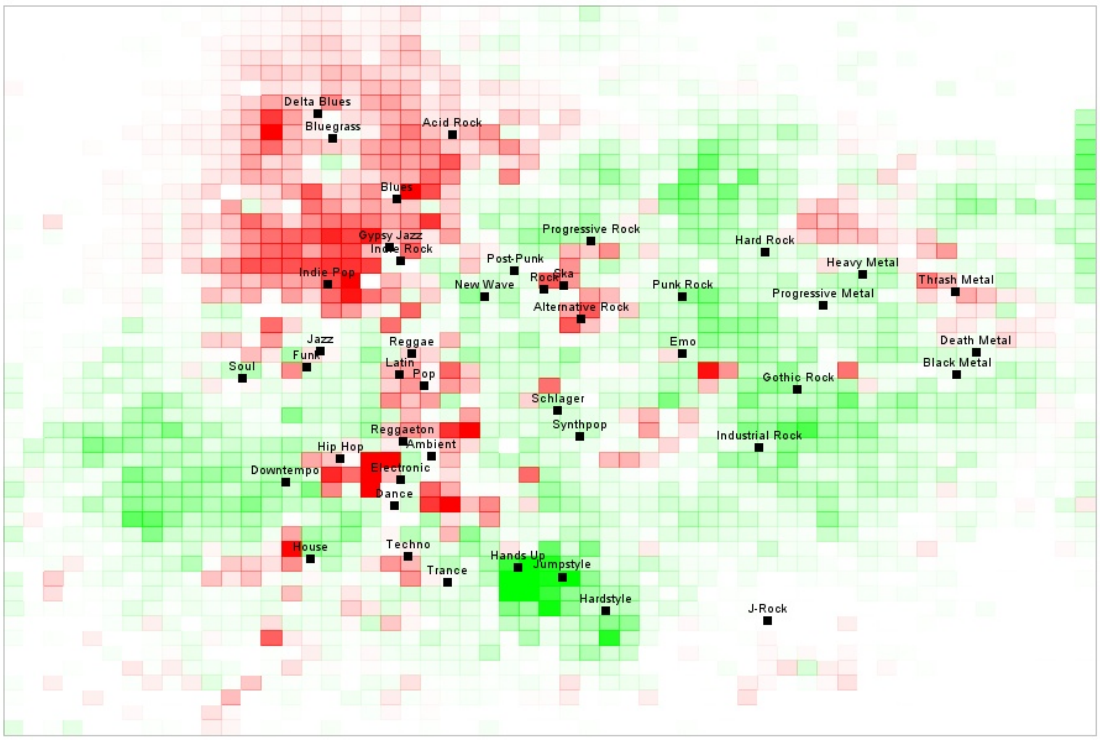
MusicExplorer PartyDJ



January 6, 2010

Adrian Waeber, awaeber@ee.ethz.ch
Daniel Waltisberg, wdaniel@ee.ethz.ch

Advisor: Michael Kuhn

Supervisor: Prof. Dr. Roger Wattenhofer

Abstract

The way how people listen to music has changed significantly in the past years, and the topic of playlist generation grew to an important topic. While many existing approaches for playlist generation are based on single users, we focus on playlist generation for a group of people, in particular when a group of people socialize at a private party.
First, users need to communicate their taste. For this task, we benefit from the possibilities of the social community Facebook and created an application where users have the possibility to define their favourite and also not liked music. Knowing these preferences, we divide the task of playlist generation in two subtasks, namely the selection of songs and the sorting of these songs in a way that there are nice transitions. For each of these two subtasks we present two approaches.
The *Map of Music* was used as similarity measurement for songs. For the selections of songs, the first approach of the song selection is based on weighting all songs in the embedding of the *Map of Music* with different gaussian distributions and the second approach focusses especially on the artists that were selected by the users. For the sorting of the selected songs, we worked in the first approach with clusters and in the second approached with an adapted version of the nearest neighbour algorithm.
For playlist verification, we built an application which uses YouTube as music source and we performed two user studies with it. This application can also be used as a player for a party.

Contents

1. Introduction

In the last years the world of music and the way how people listen to it have changed significantly. Ten years ago the main medium for music were CDs. In the last years more and more people have their music stored in a digital form on their computer or mp3-player. Nowadays, also the possibilities of Web 2.0 platforms like Facebook[1], last.fm[2] or YouTube[3] for listening to music are widely used. In this way, people do not even have to store music locally, they can, for example, just watch the video clips they like on YouTube and share the links to the clips with their friends on Facebook. Nowadays, it is not a rarity that a song on CD is sold just hundreds or thousands of times, but watched millions of times on YouTube. This shows how important alternative ways of listening to music have become.
In this thesis we want to take advantage of Web 2.0 platforms and build an application on top of it. We are interested to find new ways to select music in situations when people socialize. In particular, we focused on the scenario of a private party. Some time ago, people had to agree on a CD to play, resulting in long sequences of songs of a given artist or album. Also, the party guests often brought their own CDs to enlarge the party's music collection on the one side, and to make sure their favorite songs will be played on the other side. Nowadays, as mentioned above, the music selection is typically based on large collections of digitally stored music, which is either played in some random fashion, or by (tedious) manual selection carried out by the host or some guests. The goal of this thesis is to provide sophisticated methods for collaborative music selection that go beyond random or manual decisions, and that take the guests' preferences into account and maximizes the guests' satisfaction during the party.

In the recent research at the ETH Zürich the *Map of Music* (for details see Section 2.2) was created. In this thesis, the advantages of this existing map are combined with information from social websites like Facebook and last.fm. While the music map can provide information about music similarity, the social web can provide information about the party guests' behavior and preferences.
In particular, there are the following problems, which are solved in this thesis:

- Find ways to extract the party guests' music taste and describe it using the *Map of Music*.
- Find algorithms that (semi-)automatically create song sequences that maximize

[1] http://www.facebook.com
[2] http://www.last.fm
[3] http://www.youtube.com

the audience's satisfaction, based on the taste of the individual guests.

- Include direct feedback throughout the party to give a better control.
- Choose suitable sources of music.
- How can people be attracted to use the system?

By taking all these problems into account we come up with an application, which is called *PartyDJ* in the following.

In Chapter 2, we present relevant related work to the topic of playlist generation and group recommendation and also the existing projects which belong to the *Map of Music*.
Chapter 3 to 6 are the technical parts of the work and describe the details of the PartyDJ application and its components.
The results of two user studies through Facebook are presented in Chapter 7.

2. Related Work

2.1. General Related Work

There exist many approaches and applications around the topic of generating "intelligent" playlists. The most famous ones are probably last.fm, which provides an online music player that plays tracks the user likes (or is expected to like), and the Genius function of Apple iTunes, which creates playlists with your tracks according to its popularity measurements provided by the iTunes servers.
Another famous webapplication is the internetradio Pandora[4]. Pandora is an automated music recommendation and Internet radio service created by the Music Genome Project. Users enter a song or artist that they enjoy, and the service responds by playing selections that are musically similar. Users provide feedback on approval or disapproval of individual songs, which Pandora takes into account for future selections [7].
A last webapplication, we want to mention here, is the Music Artist Cloud[5], which provides a cloud of artists similar to the entered one and the possibility to watch YouTube videos from all of them.

In the following paragraph we mention some approaches for generating user specific playlists - and this is the main difference to our project: these (and also many other) approaches have all the idea to generate playlists for one specific user and not for a group of users like in PartyDJ.
PATS (Personalized Automatic Track Selection)[25] generates playlists that suit a particular context-of-use, that is, the real-world environment in which the music is heard (i.e. party, romantic evening, traveling,...). To create playlists, it uses a dynamic clustering method in which songs are grouped based on their attribute similarity. An inductive learning algorithm is used to reveal the most important attribute-values for a context-of-use from preference feedback of the user. In details, a user has to choose a start song and after that PATS generates and presents a playlist, which includes the selected song and songs that are similar to the selected one. While listening, a music listener indicates what songs in the playlist do not fit the intended context-of-use.
Flexer et. al [12] present an algorithm which generates playlists based on a start and end song. It has, like in PartyDJ also, the goal of smooth temporal transitions, allowing users to discover new songs in a music collection. In contrary to the *Map of Music*, this approach is based on audio similarity.

[4]http://www.pandora.com
[5]http://musicartistcloud.appspot.com/

The approach by *Andric et al.*[10] is quite similar to the one used in the *Map of Music* (see next section). It ignores metadata and instead focuses on examining the listening habits. Algorithms are presented that track the listening habits and form a so called listener model - a profile of listening habits. The listener model is then used for automatic playlist generation. Examples of tracked listening habits and consequences are: "if a track is played often a short time ago, it is expectable that it will be played in the near future" or "if a group of tracks is played together a number of times recently, it is quite likely for the whole group to be played in the same order in the near future as well".
An other approach is by *Aucouturier et al.*[16]. The playlists are generated automatically from a set of so-called global constraints, which specify properties of the whole list. Examples for such constraints are: "All Different" (the playlist should no contain the same title twice), "Duration" (it should not last more than 80 minutes), "Continuity" (the genre of a song should be close to the genre of the next song), "Progression" (the sequence should contain songs with increasing tempo, etc.). It is shown that in general the computation of playlists among such combined constraints is NP-hard.

The problem of finding results for a group and not just a single user is treated in [13]. According *Jameson et al.* the 4 subtasks of a group recommender are:

1. acquiring information about the user's preferences
2. generating recommendations
3. explaining recommendations
4. helping users to settle on a final decision

Point 4 represents a difference to our approach: we search for a whole list of songs where it is possible that sometimes one user is happy and somtimes another one. But according *Jameson et al.* the system is supposed to make recommendations concerning just one decision, e.g. watch a film or go to a restauran with a group of people. There exist several possible goals for such an application like maximizing average satisfaction, minimizing misery or ensuring some degree of fairness.
A sample application that is also treated in the thesis above is Flytrap [11], an intelligent group music recommender. Flytrap is a group music environment that knows its users' music tastes and can automatically construct a soundtrack that tries to please everyone in the room. The system works by paying attention to what music people listen to on their computers. This information can be sent to a base station in the room over RFID badges.

A last application that comes close to our project is Flycasting [17]. Flycasting stands for on the fly broadcasting and can be used for online radio stations. The goal of Flycasting is to create playlists that matches the musical preferences of an online radion station's current listeners best. As the audience changes, the type and style of songs being played should also change to match the audience's consensus tastes. It uses collaborative filtering techniques to generate a playlist in real-time based on the request histories of the current listening audience.

2.2. Map of Music

The PartyDJ project belongs to a series of master and semester thesis in the Distributed Computing Group (DCG) at the ETH Zürich which can be outlined as the *Map Of Music*, whose general idea is presented in [23]. This Euclidean map was created in previous projects and contains more than 500'000 songs. It places similar songs close to and distinct songs far apart from each other. Music similarity information is thereby derived from information from last.fm, which provides the information according to users' listening behaviour and therefore no meta tags are necessary. The *Map Of Music* exhibits several advantages in terms of applications. It allows, for example, to quickly find songs similar to each other, to define regions of interest, etc. Based on this map, the music-explorer website[6] was developed. It provides a similarity based view on music collections.
In the whole project, the source for the music, i.e. songs, artists and genres, is the embedding of the *Map of Music*. The distances between songs in the *Map of Music* are taken into account in the final application to play songs (see page 32) in a well arranged way. We used some parts of a previous project based on the *Map of Music*: *YouJuke* [20]. YouJuke is a webapplication to play songs from the *Map of Music*. For doing this it uses the YouTube API and plays the videso from YouTube. You can see how it works on the official webpage[7].

Other papers belonging to the *Map of Music* are:

- *Mobile Music Explorer* [24], a mobile application, which allows users to create playlists by specifying trajectories onthe map and to use similarity based search methods to navigate through their personal music collections.
- Alternative exploration schemes for mobile devices, taking advantage of the high-dimensional music similarity space. In [18] an Android based prototye application is presented for the approaches of visual and acoustic navigation.

Based on these two papers, the newest mobile application *museek* for Android devices was developed and can be downloaded on the official webpage[8] (actual version: v0.915, December 2009). At the first glance, *museek* looks like an ordinary music player, but it provids some entirely new ways of interacting with a music collection:

- Browsing through album covers in two or three dimensions.

[6]http://www.musicexplorer.org
[7]http://www.youjuke.org
[8]http://www.museek.ethz.ch

- A smart shuffling mode remembers skipped tracks and can thus avoid not liked music styles.
- Automatically generated tag-clouds allow a fine-grained selection of the music one wants to play.
- A novel search mode is able to search similar artists.

3. PartyDJ Architecture

With the PartyDJ project it is possible to celebrate private parties, on which music, that the guests should mostly like, is (semi-) automatically played. To realize such an application, different steps are needed. In our approach we come up with three steps:

1. Collect data about users' preferences of music and store it to a database.
2. Bring these input together and generate playlists that maximize the users' satisfaction.
3. Play the songs during the party.

To connect these three steps, data has to be stored in one step and reloaded in the following one. We use a database on the servers from the Distributed Computing Group at the ETH Zürich to do this.
In all three steps the existing *Map of Music* (see Section 2.2), which provides information about music similarity, is used. The database of the songs in the map is also called embedding. In the first step, the music preferences which can be selected by the users are the songs, artists and genres which are available in the embedding. In the second step the similarity of songs in the map is used to generate playlists, which are finaly played in third step.

The whole application is built with the Google Web Toolkit (GWT) [3], a development toolkit for building and optimizing complex browser-based applications. With this toolkit most of the programming work can be done in Java.

According the three mentioned steps, the whole PartyDJ project consists of three main parts, which are shortly described in the following as an overview. For a more detailed description of each component see chapters 4, 5 and 6.

3.1. Facebook Application

To collect data about the preference of music of the users, who will join the party, we decided to use Facebook due to its growing popularity. It's an obvious approach to use the aid of such a famous social community to come up with an application for a situation when people socialize. Therefore, we created on Facebook the application

"MusicExplorer PartyDJ"[9], which can be used if you are loged in to Facebook. There exist different methods for the users to submit their preferences (see Section 4.2). All the inputs by the users are stored to our database. Additionally, the Facebook application provides the possibility to create and join parties (see Section 4.5.2) and to get visual representation of the *Map of Music* (see Section 4.5.3).

3.2. Playlist Generation

When the data is collected, it can be loaded from our database and different algorithms are used to process the user inputs and to rate them (see Section 5.1). From the subset of songs with high ratings, two different transition techniques (see Section 5.2) are used to generate playlists, which will also be stored in our database.

3.3. DJ Player

Finally, during the party only this component is needed. The player is based on the existing project YouJuke[10] and allows to play the songs from the generated playlist, which are loaded from our database. To be able to do this, YouTube is used as music source and videos are included to the DJ Player. It is also possible to add songs manually during the party to improve user satisfaction.

[9] http://apps.facebook.com/musicexplorerpartydj
[10] http://www.youjuke.org

4. Facebook Application

4.1. Integration to Facebook

Nowadays, many people are present in social networks and many parties are published through them, in particular through Facebook. We decided to build the application for Facebook in order to benefit from the possibilities of such a network. With having the application in Facebook, it is simpler to invite other users for a party and to advertise the application to a bigger audience.
Also the storage of user specific data is quite trivial because every Facebook user has a unique ID. So all needed data for a user can be stored into the PartyDJ database according to this ID, no additional registration procedure is needed.
The only disadvantage of Facebook is that there exists no really good Java API that can be integrated into the Google Web Toolbox. We decided to handle the authentification process via PHP, as described in [9].

4.2. Collecting User Data

The main functionality of the Facebook application is to collect data about the users' taste of music. In particular, the users can say if they like oder dislike songs, artists and genres. This can be done in three ways.

1. The users can give their opinion about randomly presented songs, artists and genres from the embedding. There are only songs and artists presented which have a minimum value of popularity (this information is taken from last.fm). The different music genres are chosen according to [6] and [1] and some modern genres like jumpstyle or hands up, for example, are added manually.
2. It is also possible to manually add songs, artists and genres, with the restriction that they have to be already present in the embedding. Otherwise there exist no coordinates in the *Map of Music*.
3. The fastest and simplest way of adding data is made for users who have a last.fm profile. They can simply import songs and artists from their scrobbled data and decide, which ones they would like to listen to at the party. In contrary to the

second method, with this one it is also possible to add songs which are not yet present in the embedding. How this can be done is described in Section 4.3.

All inputs of the users are stored to our database according to his unique Facebook ID and are reloaded when he logs in another time.

4.3. Guessing Coordinates of New Songs and Artists

If a user wants to add a song or an artist from his last.fm profile, which is not yet present in the embedding, this can be done in the following way.
For an unknown artist, a request will be sent to the last.fm API [2] to get similar artists. Based on these the coordinates for the unknown artist in the 10 dimensional space are calculated as the mean values of the coordinates of the similar ones. If none of the similar artists is in the embedding, then the unknown artist can't be added.
Unknown songs receive just the coordinates of their artist, so they will be put in the center of mass of the artist.
Because of the ongoing process of crawling data from last.fm (as mentioned on page 11), coordinates of songs and artists from last.fm can easily be guessed. But if the songs or artists are that new, that they are not yet crawled, then it is actually not possible to add the song to the embedding.

4.4. 2-Dimensional Representation of Music

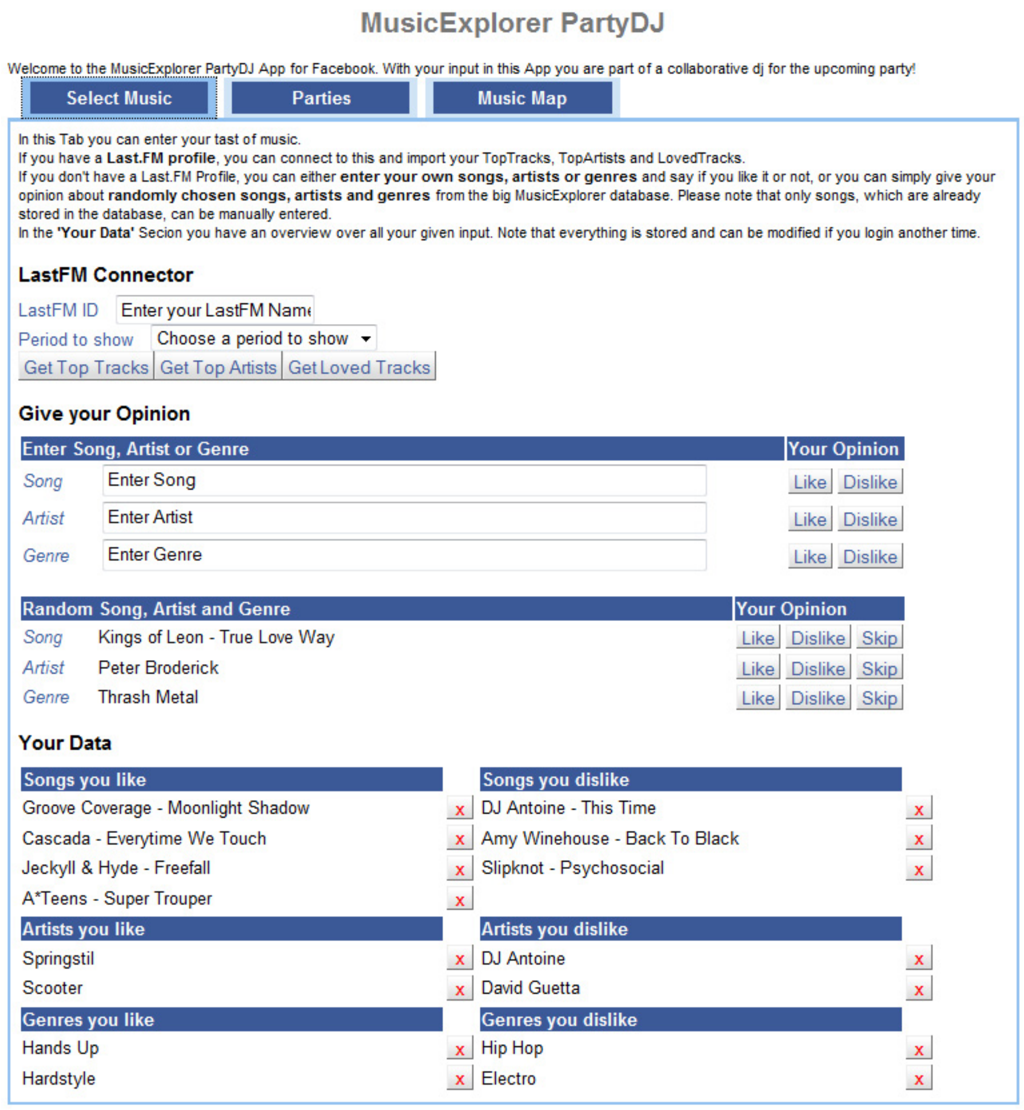
The problem of highdimensional data is always its representation for human users. During the event "Nacht der Forschung"[11], another project that uses the *Map of Music* was presented. For a user friendly visualisation, a two dimensional representation was needed. A principal component analysis with the centroids of the most relevant genres was used and the songs were distributed according to the distances to them.
For our application, we used this two dimensional representation to visualise the user inputs (see Section 4.5.3), the *Map of Music* after the gaussian weighting (see Section 5.1.1) and after the clustering (see Section 5.2.1), and to visualise the playlist transitions (see Section 5.2.3).

[11]An evening in Zürich at which the universities show current research projects to the public, see http://www.nachtderforschung.ch

4.5. User Interface

The user interface of the Facebook application consists of the three tabs "Select Music", "Parties" and "Music Map".

4.5.1. Select Music



MusicExplorer PartyDJ

Welcome to the MusicExplorer PartyDJ App for Facebook. With your input in this App you are part of a collaborative dj for the upcoming party!

In this Tab you can enter your tast of music.
If you have a **Last.FM profile**, you can connect to this and import your TopTracks, TopArtists and LovedTracks.
If you don't have a Last.FM Profile, you can either **enter your own songs, artists or genres** and say if you like it or not, or you can simply give your opinion about **randomly chosen songs, artists and genres** from the big MusicExplorer database. Please note that only songs, which are already stored in the database, can be manually entered.
In the **'Your Data'** Secion you have an overview over all your given input. Note that everything is stored and can be modified if you login another time.

LastFM Connector

LastFM ID: Enter your LastFM Name

Period to show: Choose a period to show

Give your Opinion

Enter Song, Artist or Genre		Your Opinion
Song	Enter Song	Like Dislike
Artist	Enter Artist	Like Dislike
Genre	Enter Genre	Like Dislike

Random Song, Artist and Genre		Your Opinion
Song	Kings of Leon - True Love Way	Like Dislike Skip
Artist	Peter Broderick	Like Dislike Skip
Genre	Thrash Metal	Like Dislike Skip

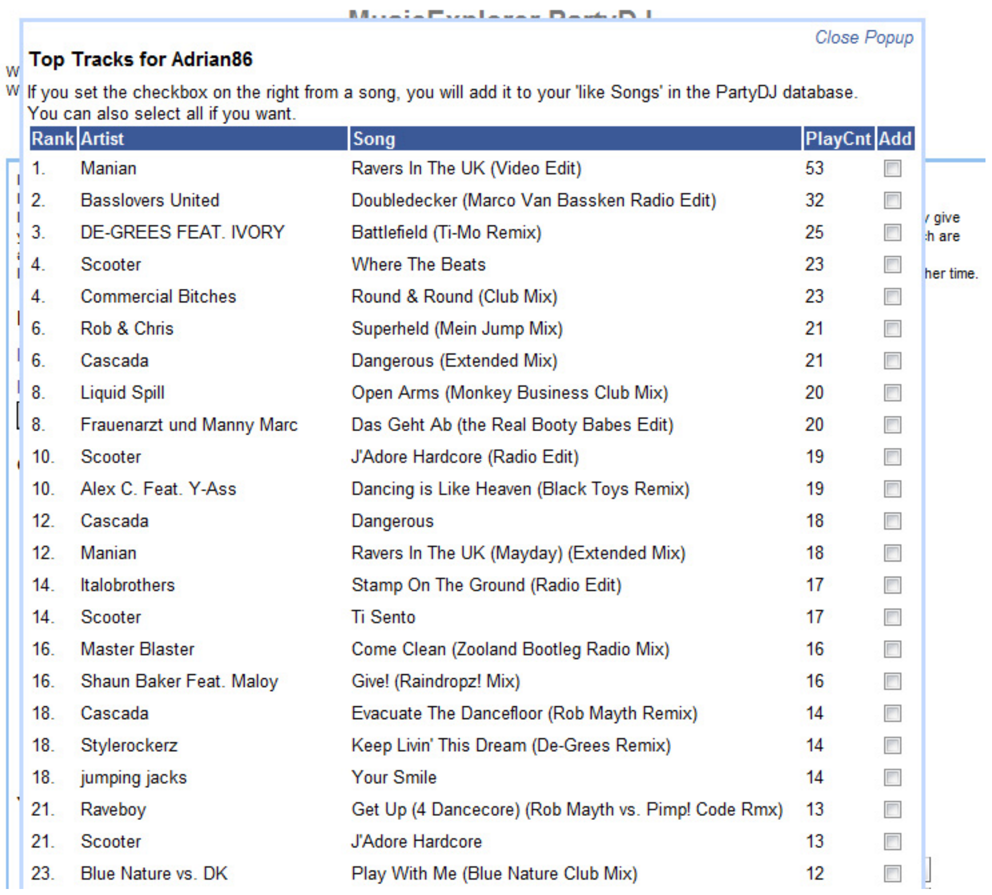
Your Data

Songs you like		Songs you dislike	
Groove Coverage - Moonlight Shadow	x	DJ Antoine - This Time	x
Cascada - Everytime We Touch	x	Amy Winehouse - Back To Black	x
Jeckyll & Hyde - Freefall	x	Slipknot - Psychosocial	x
A*Teens - Super Trouper	x		
Artists you like		**Artists you dislike**	
Springstil	x	DJ Antoine	x
Scooter	x	David Guetta	x
Genres you like		**Genres you dislike**	
Hands Up	x	Hip Hop	x
Hardstyle	x	Electro	x

Figure 4.1.: Select music tab in the Facebook application

If the user is logged in to Facebook and the application is loaded, the application looks like in Figure 4.1.

In the "Give your Opinion" part the users can manually enter songs, artists and genres and tell if they like or dislike it. They can also do this for randomly chosen items, with the possibility to skip it if they do not know it or do not want to rate the entry. The manual input is handled with suggestion boxes based on the data in the embedding. The limitation is, that the users can just enter songs, artists and genres which are present in the embedding, as mentioned in Section 4.2.
In the "Your Data" part all the liked and disliked entries of users are stored and loaded everytime they log in to Facebook. Here it is also possible to delete the entries if the users have changed their opinion about one.



Top Tracks for Adrian86

If you set the checkbox on the right from a song, you will add it to your 'like Songs' in the PartyDJ database.
You can also select all if you want.

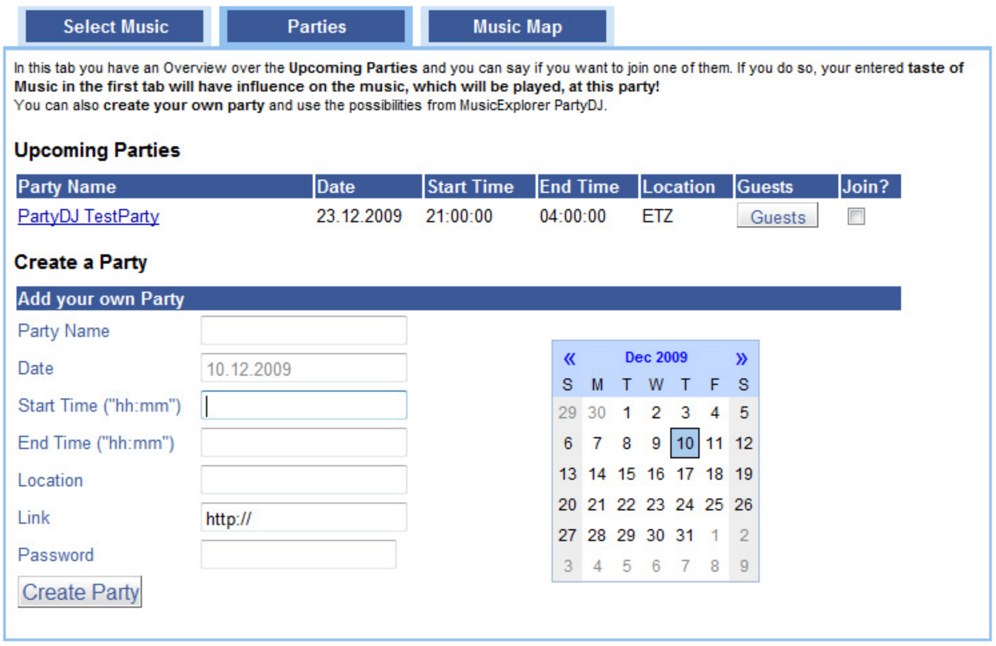
Rank	Artist	Song	PlayCnt	Add
1.	Manian	Ravers In The UK (Video Edit)	53	☐
2.	Basslovers United	Doubledecker (Marco Van Bassken Radio Edit)	32	☐
3.	DE-GREES FEAT. IVORY	Battlefield (Ti-Mo Remix)	25	☐
4.	Scooter	Where The Beats	23	☐
4.	Commercial Bitches	Round & Round (Club Mix)	23	☐
6.	Rob & Chris	Superheld (Mein Jump Mix)	21	☐
6.	Cascada	Dangerous (Extended Mix)	21	☐
8.	Liquid Spill	Open Arms (Monkey Business Club Mix)	20	☐
8.	Frauenarzt und Manny Marc	Das Geht Ab (the Real Booty Babes Edit)	20	☐
10.	Scooter	J'Adore Hardcore (Radio Edit)	19	☐
10.	Alex C. Feat. Y-Ass	Dancing is Like Heaven (Black Toys Remix)	19	☐
12.	Cascada	Dangerous	18	☐
12.	Manian	Ravers In The UK (Mayday) (Extended Mix)	18	☐
14.	Italobrothers	Stamp On The Ground (Radio Edit)	17	☐
14.	Scooter	Ti Sento	17	☐
16.	Master Blaster	Come Clean (Zooland Bootleg Radio Mix)	16	☐
16.	Shaun Baker Feat. Maloy	Give! (Raindropz! Mix)	16	☐
18.	Cascada	Evacuate The Dancefloor (Rob Mayth Remix)	14	☐
18.	Stylerockerz	Keep Livin' This Dream (De-Grees Remix)	14	☐
18.	jumping jacks	Your Smile	14	☐
21.	Raveboy	Get Up (4 Dancecore) (Rob Mayth vs. Pimp! Code Rmx)	13	☐
21.	Scooter	J'Adore Hardcore	13	☐
23.	Blue Nature vs. DK	Play With Me (Blue Nature Club Mix)	12	☐

Figure 4.2.: TopTracks for a period of 3 months from Last.FM

With the "LastFM Connector" part the application gives a possibility to last.fm users to

collect data from their profile. In particular they can load their LovedTracks, TopTracks or TopArtists (these are songs and artist, which the user mostly listened to it) for a chosen period (overall, last year, last 6 months, last 3 months, last week). Received TopTracks, for example, can look like in Figure 4.2.
The users can now check, which tracks they want to add to their liked songs in the Facebook application.

4.5.2. Parties



In this tab you have an Overview over the **Upcoming Parties** and you can say if you want to join one of them. If you do so, your entered **taste of Music in the first tab will have influence on the music, which will be played, at this party!**
You can also **create your own party** and use the possibilities from MusicExplorer PartyDJ.

Upcoming Parties

Party Name	Date	Start Time	End Time	Location	Guests	Join?
PartyDJ TestParty	23.12.2009	21:00:00	04:00:00	ETZ	Guests	☐

Create a Party

Add your own Party

Party Name	
Date	10.12.2009
Start Time ("hh:mm")	
End Time ("hh:mm")	
Location	
Link	http://
Password	

Create Party

« Dec 2009 »

S	M	T	W	T	F	S
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Figure 4.3.: Create and join parties

In the upper part of the second tab called "Parties" the details of all upcoming parties are shown. Here users can see, who is joining a specific party and they can check, if they plan to join this party themself. If they do so, their inputs in the first tab will take into account for the playlist generation (see chaper 5) for this party.
In the lower part the users have the possibility to create their own party. For doing this, they have to give the party a name, pick a date, give start and end time in correct format, enter a location and optionally enter a link to a website. Finally, they also have to define a password, which will be needed to load the playlists and all data for the party in the DJ Player (see Section 6.2.1).

4.5.3. Map of Music

In the last tab, the users can see where their liked (green) and disliked (red) songs are placed in a 2-dimensional representation of the *Map of Music* (see Figure 4.4). To show, where the different music genres are placed, the centers of mass from the most popular genres in the PartyDJ database are shown in the lower picture.
The brightness of the different parts in the map represents the density of songs at these points. This means that in bright parts of the map are much more songs placed than in darker parts.

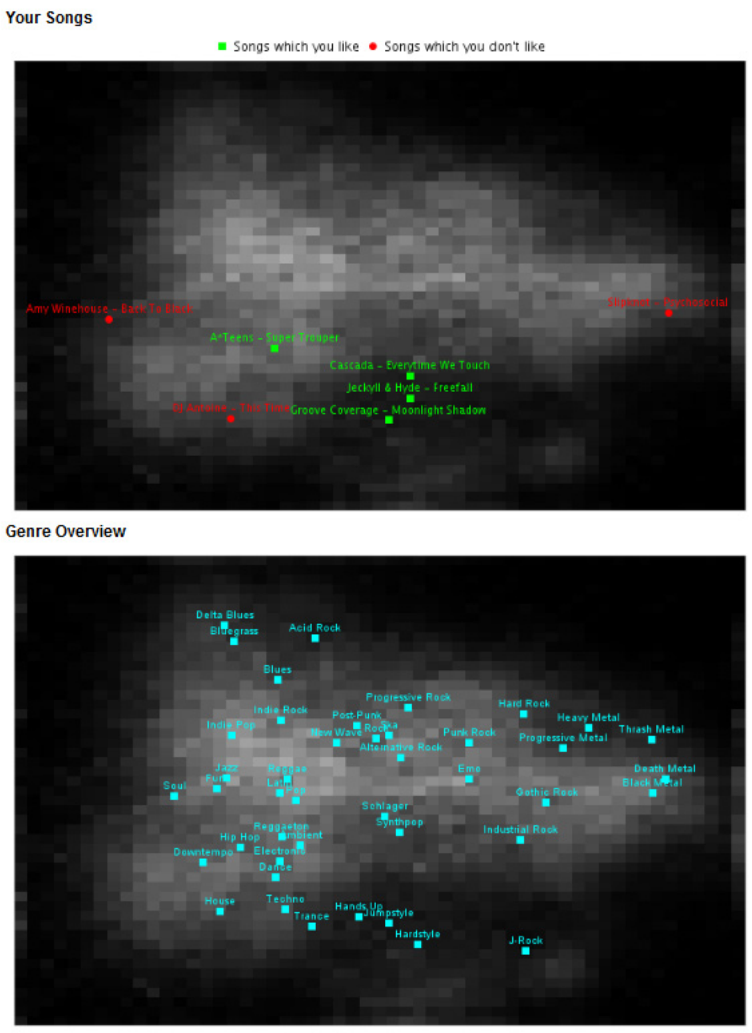


Figure 4.4.: Liked and disliked songs in a 2-dimensional representation

5. Playlist Generation

From the user input, we get three different kinds of music information which are used to specify the music taste of a user, namely songs, artists and genres that the user likes or dislikes.
The final goal of the playlist generation is to create a playlist which pleases most of the people. This is hard, especially if the group is mixed with people with different music tastes. Imagine for example that half of the users likes "Hip Hop" and the other half prefers "Trance". Rahter trying to find songs which are very liked by both groups, we choose the most liked songs by each group and accept, that there will be songs which are not liked by all people. But it can then be switched between these music styles and the overall satisfaction should be maximized.
The task of playlist generation can therefore be subdivided into two tasks. First, we have to process the user inputs and select those songs which correspond to the wishes of the user. This task is done with two different approaches in Sections 5.1.1 and 5.1.2. Secondly, we have to arrange the songs in such a way, that the transitions are nice, but that there is still enough variety during the playlist. Two approaches that are trying to fullfil this transistion requirement are discussed in Sections 5.2.1 and 5.2.2.

5.1. Processing of User Input

The first approach of user input processing is based on weighting all songs in the embedding of the *Map of Music* with different gaussian distributions. The idea of the weighting based on gaussian distributions is, that songs, which are similar, are near to each other in the *Map of Music* (see Section 2.2) and should therefore receive a positive weight. After the weighting process, the songs with the highest weights are selected and used for the transition task.
The second approach is based on the main result of the previous one which states that the selected artists play an important role in the selection of the playlist songs.

5.1.1. Gaussian Distribution

We have a set of songs $S = (s_1, s_2, \ldots, s_L)$, a set of artists $A = (a_1, a_2, \ldots, a_M)$ and a set of genres $G = (g_1, g_2, \ldots, g_N)$ which were selected by the users. For each artist a we

have a set of songs S_a which were produced by this artist and for each genre g we have a set of songs S_g which were tagged by the genrename.
The *Map of Music* provides us 10-dimensional coordinates $\vec{x}_i = (x_{i1}, \dots, x_{id})^T$ for each song i in the embedding. Each song i will be weighted by the songs, artists and genres that were selected by the users. For a song $s_l \in S$, an artist $a_m \in A$ and a genre $g_n \in G$, the weighting functions of a song i in the embedding are denoted as $v_{s_l}(i)$, $v_{a_m}(i)$ and $v_{g_n}(i)$. The final combination of these weights is discussed in Section 5.1.1.6.

5.1.1.1. Visualisation

At each weighting process, we used the two dimensional representation described in Section 4.4 to visualise the weigth distribution among all songs in the embedding. Each rectangle consists of the sum of all distributed weights in that region. Regions with positive values are green and regions with negative values are red. As user input we selected all songs, artists and genres which were registered for our applications.

5.1.1.2. Fairness

To guarantee fairness between the users, the contribution of weights by each user is limited. If the maximal contribution of one user is reached, the weights are distributed uniformly among all selections off this user.
For each song $s_l \in S$, each artist $a_m \in A$ and each genre $g_n \in G$, this yields to a weight of p_{s_l}, p_{a_m} and p_{g_n}.

5.1.1.3. Weighting based on Songs

To describe a music taste, the most precise way is to use a set of songs which the user likes or dislikes, but because of the large number of required songs it is also quite time consuming for the user. Based on the basic assumption that similar songs lie near to each other in the *Map of Music*, not only the selected song are weighted, but also the songs which are similar to the chosen one. The *Map of Music* gives a measure of similarity when looking at the distance between the songs.
For each song $s_l \in S$, each song i in the embedding is weighted according to a Gaussian distribution

$$\hat{v}_{s_l}(i) = p_{s_l} \exp\big(-\tfrac{1}{2\sigma^2} |\vec{x}_i - \vec{x}_{s_l}|^2 \big) \tag{5.1}$$

with fixed variance σ^2.
For faster computation, a kd-tree[12] was used to determine the neighboorhood N_{s_l} of

[12]See http://en.wikipedia.org/wiki/Kd_tree

songs within distance d_{min}. The distance d_{min} has been set to the distance at which the weight of a song i would be lower than 1% compared to the maximum value, that is

$$d_{min} = \sigma\sqrt{2\ln(100)}. \tag{5.2}$$

For all songs i outside N_{s_l}, no weighting is applied, that is to say

$$v_{s_l}(i) = \begin{cases} \hat{v}_{s_l}(i) & \text{if } i \in N_{s_l} \\ 0 & \text{else} \end{cases} \tag{5.3}$$

Additionally, we limit the sum of distributed weights by a single song $s_l \in S$ to a fixed amount. Starting at the nearest song to the center, we stop as soon as the sum of the previously distributed weights exceeds the limit. This restriction is needed because there exist songs which lie at the very same position. If one wouldn't limit the weightening, all songs at the position would get the maximal weights and this would be unfair to other songs.
The weighting result based on the user selected songs is visualised in Figure 5.1.

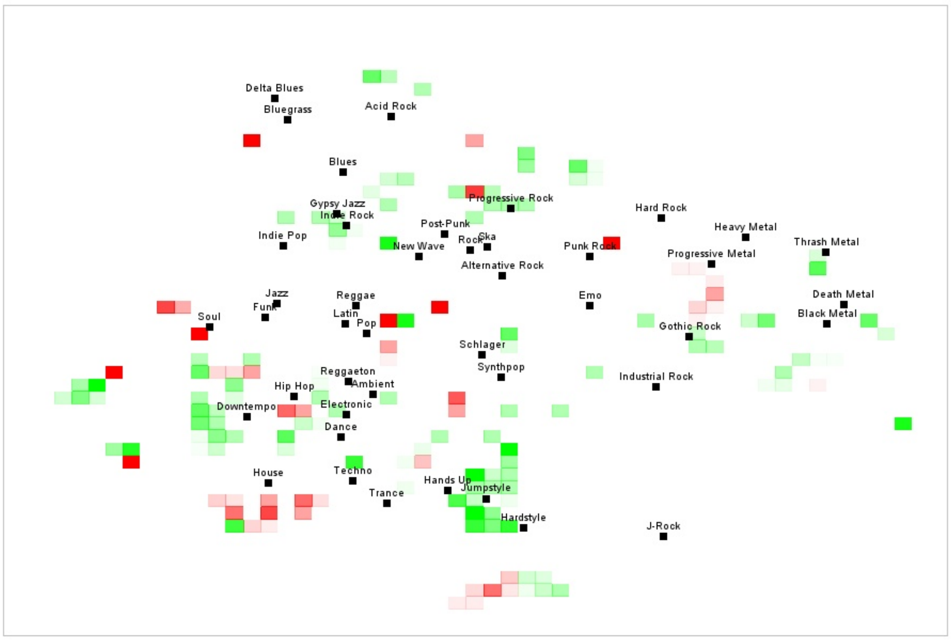


Figure 5.1.: The *Map of Music* after the weighting with the user selected songs.

5.1.1.4. Weighting based on Artists

Using a specific artist, we are able to address a big number of songs at once. The idea of weighting according to artists instead of single songs is driven by the fact that if one

likes a song of an artist, one often likes the other songs of the artist as well.
For the weighting process of an artist $a_m \in A$, we select the set of songs S_{a_m} produced by this artist and weight them according to the artist weight p_{a_m}. This yields to a weighting of

$$v_{a_m}(i) = \begin{cases} p_{a_m} & \text{if } i \in S_{a_m} \\ 0 & \text{else} \end{cases} \tag{5.4}$$

for a song i and a specific artist a_m.
The weighting result based on user selected artists is visualised in Figure 5.2.

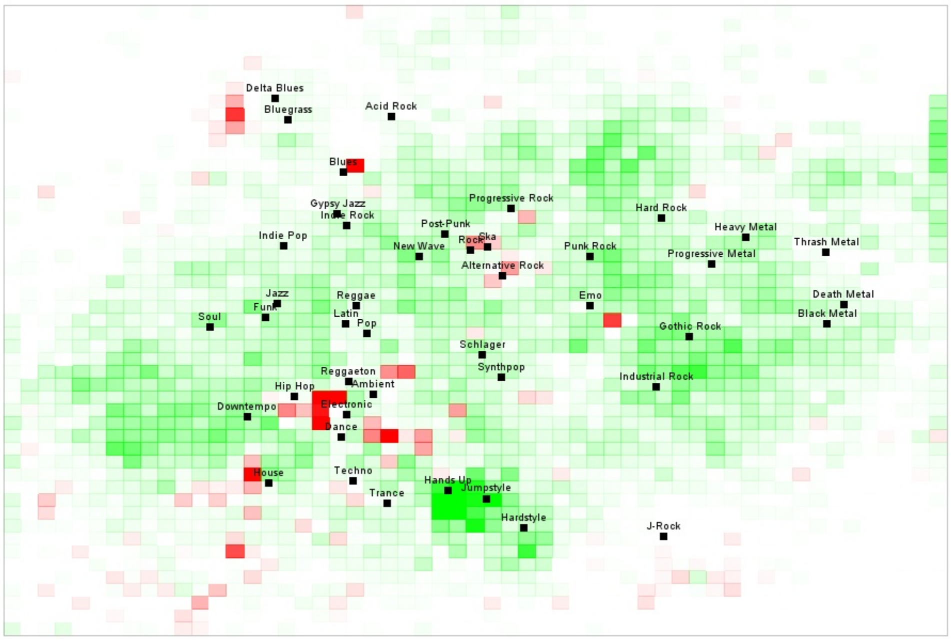


Figure 5.2.: The *Map of Music* after the weighting with the user selected artists.

5.1.1.5. Weighting based on Genres

For the weighting based on genres, we used the additional information of tag names which are avaible through last.fm. For a specific genre $g_n \in G$, we have a set of songs S_{g_n} which contained a tag which matches with the name of the genre. From this collection of songs, we built a 10-dimensional gaussian distribution $N(\vec{\mu}_{g_n}, \Sigma_{g_n})$ with expectation

$$\vec{\mu}_{g_n} = \frac{1}{|S_{g_n}|} \sum_{\vec{x}_g \in S_{g_n}} \vec{x}_g = \begin{pmatrix} \mu_{g_n 1} \\ \vdots \\ \mu_{g_n d} \end{pmatrix} \tag{5.5}$$

and the 10×10 dimensional covariance matrix

$$\Sigma_{g_n} = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \cdots & \sigma_{dd} \end{bmatrix}, \text{where} \tag{5.6}$$

$$\sigma_{kl} = \frac{1}{|S_{g_n}|} \sum_{\vec{x} \in S_{g_n}} (x_k - \mu_{g_{nk}})(x_l - \mu_{g_{nl}}). \tag{5.7}$$

This yields to the the scaled density function

$$f_{g_n}(\vec{x}) = \frac{1}{|\Sigma_{g_n}|^{1/2}} \cdot \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_{g_n})^T \Sigma_{g_n}^{-1} (\vec{x} - \vec{\mu}_{g_n})\right). \tag{5.8}$$

For normalisation and fairness reasons, we introduce a genre normalisation constant c_{g_n} for each genre g_n. The genre constants are chosen in such a way that the distributed weights among all songs in the embedding E is equal for all genres, that is

$$c_{g_n} = \left(\sum_{\vec{x} \in E} f_{g_n}(\vec{x}) \right)^{-1}. \tag{5.9}$$

For each genre g_n in the set G of selected genres, all songs in the embedding are weighted according to the gaussian distribution $N(\vec{\mu}_{g_n}, \Sigma_{g_n})$ and with weight $c_{g_n} \cdot p_{g_n}$. For a song i with coordinates $\vec{x}_i$ and a genre g_n, this yields to a weighting according to

$$v_{g_n}(i) = c_{g_n} \cdot p_{g_n} \cdot f_{g_n}(\vec{x}_i). \tag{5.10}$$

The weighting result based on user selected genres is visualised in Figure 5.3.

5.1.1.6. Combination

For each song i in the embedding, the total weight is computed according to

$$v(i) = w_s \cdot \sum_{s_l \in S} v_{s_l}(i) + w_a \cdot \sum_{a_m \in A} v_{a_m}(i) + w_g \cdot \sum_{g_n \in G} v_{g_n}(i) \tag{5.11}$$

The hard part of the merging of the different evaluation schemes is to find the parameters w_s, w_a and w_g in such a way, that each part of information has an useful influence on the final set of songs.
For an estimation of a good set of parameters, crossvalidation was used. For the crossvalidation, 10% of the positive user selected songs were removed. For a fair comparison between different weights, we calculate the average value of the removed songs after the evaluation and divide it with the songvalue required for entering the playlist with

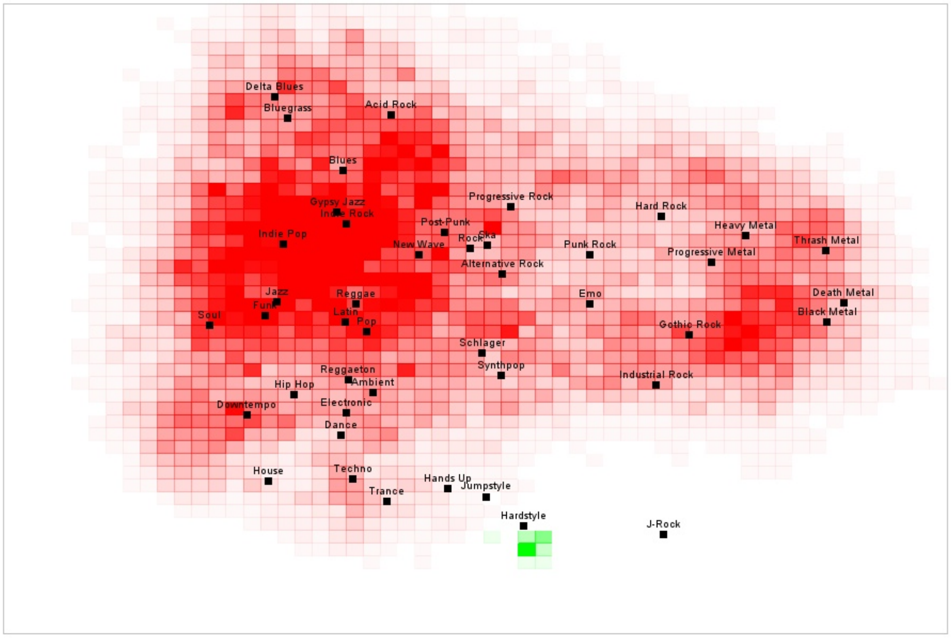


Figure 5.3.: The *Map of Music* after the weighting with the user selected genres.

the top100 songs. We call this factor efficiency, because it shows how good the algorithm performs. This efficiency calculation was repeated for different song, artist and genre weights and averaged for different removed songs.
In the following figures, the influece of the weights is displayed. In Figure 5.4, the song weight against efficiency for different genre weights and fixed artist weight is plotted. In Figure 5.5 and Figure 5.6, the influece of the artist weight against efficiency for different parameters is plotted.

From these figures, the following can be concluded:

- Comparing the efficiency for song and genre weighting, the weight of the genre weighting seems to have no useful influence.
- Looking at the efficiency for artist weighting, the efficiency yields to best results when maximizing the weight of the artist weighting.

This leads to the assumption that the artist evaluation is the best tool to create good playlists. An approach that is based on this result is presented in the next section.
Because there is no maxima in the plots, we select parameters at which the efficiency is high and each weighting has some influence. The two dimensional representation of the weighting with weights $w_s = 5$, $w_a = 10$ and $w_g = 3000$ is presented in Figure 5.7.

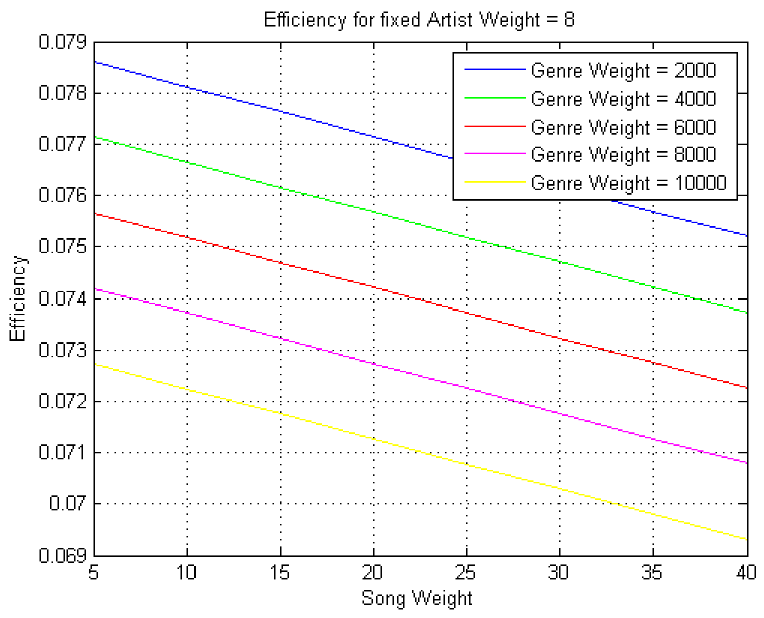


Figure 5.4.: Song weight against efficiency for different genre weights and fixed artist weight

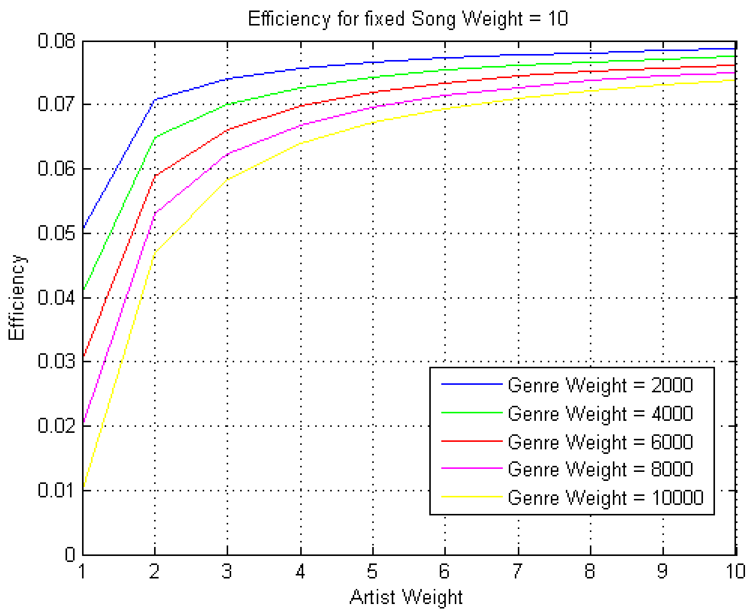


Figure 5.5.: Artist weight against efficiency for different genre weights and fixed song weight

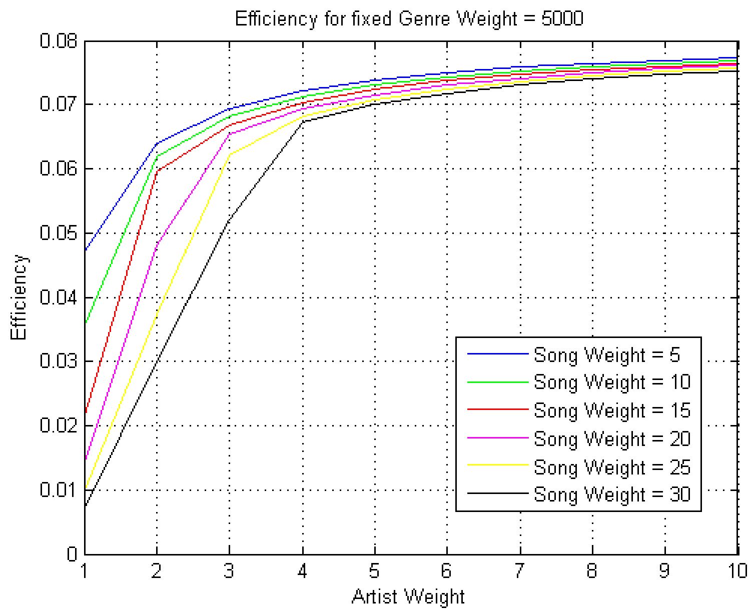


Figure 5.6.: Artist weight against efficiency for different song weights and fixed genre weight

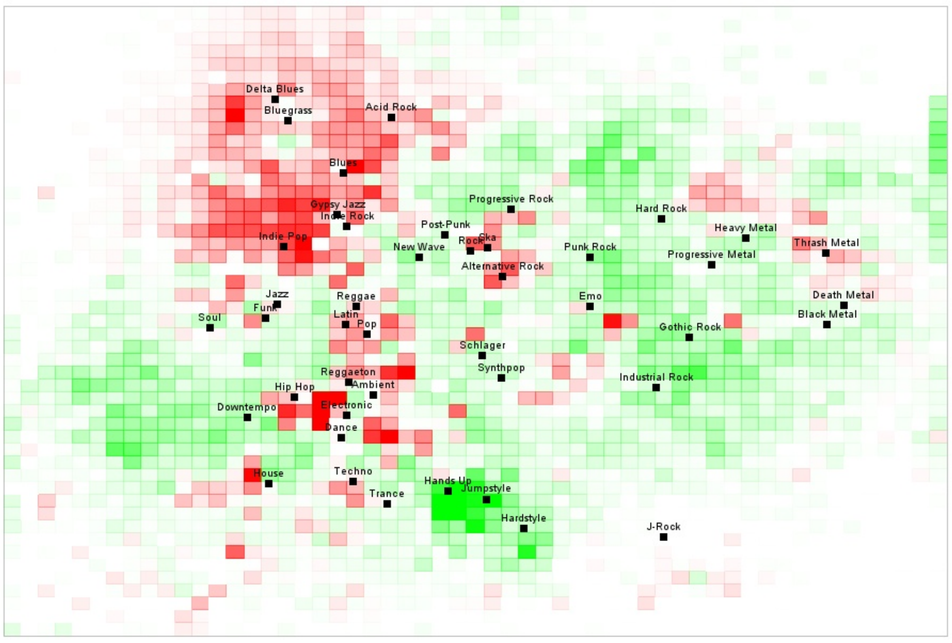


Figure 5.7.: The *Map of Music* after the weighting with user selected songs, artists and genres.

5.1.2. Artist Distribution

From the conclusions of the previous section, a new songselection method was implemented to find a list of songs which should be used for the transition step. The selection of the songs is done in the following steps:

1. Add songs that were selected by the users to the list. The number of songs per user that is added to this list is limited inverse proportional to the number of users that partizipate at the party.
2. From the set of artists that were either selected by the users or belonged to the songs that the users selected, pick a random artist. From all songs of this artist, choose a random song that has at least some minimal popularity and add it to the list.
3. Repeat step 2 until wished size of list is reached.

We added also an additional constrait, so that the number of songs by the same artist is limited.

5.2. Transitions

After the user input processing, we know the songs that should be played in the playlist to satisfy the users. The next step is to order the songs in such a way, that there is a good mixture of variety and locality. The transitions should be smooth, but it should also be avoided to stay too long in one music region. We use the coordinates from the *Map of Music* as the measure of similarity.
The first approach to this task is to look out for clusters in the selected playlists. With these clusters, we can control the time that we want to stay in each cluster and much more. This approach is discussed in section 5.2.1.
The second approach is based on the similarity between our problem and the travelling salesman problem (TSP) [8]. The TSP is the task of visiting a list of cities in such a manner that each city is visited once and the overall travelled distance is minimal. A simple solution to the TSP is the nearest neighbour algorithm (also called greedy algorithm). Implementing the nearest neighboor algorithm for our task would lead to smooth transition, but there would be very small variety. Therefore we adapted the algorithm as discussed in section 5.2.2.

5.2.1. Clustering

The task of clustering arises in many applications like machine learning, data mining, pattern recognition or image analysis. A very popular approch is the K-Means algorithm by *Stuart P. Lloyd* as discussed in [19]. A broad discussion of clustering algorithms can be found in [14], but we will use the K-Means algorithm because of its simplicity.

5.2.1.1. K-Means algorithm

The K-Means algorithm, adapted for songs in the *Map of Music*, proceeds by selecting K initial songs as cluster centers and then iteratively do the following:

1. Assign each song to its closest cluster center
2. Replace the center of each cluster by the mean of the coordinates of its assigned songs

The algorithm converges when there are no further changes in the assignment of the songs to the clusters.
There are two shortcomings with the K-Means Algorithm. First, the number of clusters K has to be estimated. Second, the algorithm converges to local minimas and there is no guaranty that this is a global minima. Additionally, we have to note that although the initial songs can be chosen arbitrarily, the algorithm is fully deterministing, given the staring centers.
To overcome these shortcomings, we run the algorithm multiple times with different K value and different initial songs and use cluster validity methods to find the best clustering. We used the Davies-Bouldin index. Other Cluster validity methods are discussed in [21] and [22].
The Davies Bouldin index is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The within-cluster dispersion d_i of a cluster C_i can be measured by the average distance of the songs associated with the cluster and the cluster center μ_i:

$$d_i = \frac{1}{|C_i|} \sum_{y \in C_i} d(\mu_i, y) \tag{5.12}$$

with $d(x, y)$ the distance between the coordinates x and y.
Between two clusters C_i and C_j, the similartiy measure R_{ij} is the sum of the the within-cluster dispersion over the distance between the two centers μ_i and μ_j, see

$$R_{ij} = \frac{d_i + d_j}{d(\mu_i, \mu_j)} \tag{5.13}$$

From this, the DB index is defined as

$$DB = \frac{1}{K} \sum_{i=1}^{K} R_i, \text{ where} \tag{5.14}$$

$$R_i = \max\{R_{ij} | 1 \leq j \leq K, i \neq j\} \tag{5.15}$$

To avoid having too many small clusters, the clustering with the best DB is postprocessed in such a way, that too small and low rated clusters are split up and the songs distributed to the nearest cluster center.

5.2.1.2. Playlist Generation

From these clusters, the playlist is generated with the following procedure:

1. Set the number of songs per walk through of each cluster proportional to the number of songs in the corresponding cluster.
2. Set a randomly chosen song of a randomly chosen cluster as start song.
3. Add songs of the current cluster in a greedy manner.
4. Select all those clusters that have been visited the fewest. Choose the nearest cluster of them as the next cluster.
5. Make a smooth transition from the current song to the center of the next cluster by adding songs lying inbetween.
6. Go to step 3 until the expected playlistsize is reached.

The fifth step needs some more explanation. All songs of the embedding were loaded into a kd-Tree [4] and the path from the start coordinates to the end coordinates was divided into multiple positions between them. The number of positions was set proportional to the distance. For each position, the surrounding songs were selected and the one with the highest popularity value was added to the playlist.

5.2.2. Travelling Salesman

Another way of generating smooth transitions is to implement the nearest neighbourhood algorithm. The nearest neighbour algorithm choses the nearest unvisited city as next move. For N cities randomly distributed on a plane, the algorithm yields on average to a length of $\mathrm{l} = 1.25 \cdot \mathrm{exact\_shortest\_length}$, which is remarkably good. It has also to be noted, that the start and endpositions have not to be the same, which yields to a further improvement. Of course, there exist many specially arranged city distributions which make the algorithm make a very bad route decision, as described in [15]. For our setting, the simplicity of the algorithm weights out its shortcomings.
The implemention of a solution of the travelling salesman problem would lead to very smooth transitions, but probably not enough variety. Therefore, we extend the nearest neighbourhood algorithm, that is

1. Select a number of songs randomly from the remaining songs
2. Perform the nearest neighbourhood algorithm through these songs
3. Go to step 1 until all songs are added

This algorithm leads to quite smooth transitions and enough variety.

5.2.3. Examples

To visualise the cluster results and the difference between the two playlist creations algorithms, we used the two dimensional representation of the *Map of Music* described in Section 4.4. As input, we used the top100 songs, which were the result of the gaussian distribution with all user inputs.
In Figure 5.8, the songpositions belonging to different clusters are displayed. It shows the clustering corresponding to the best DB index. Some of the clusters seem to be improvable, but it has to be noted that the clustering itself is done in the 10 dimensional space. Therefore it is not a surprise that the clustering seems a bit strange in the two dimensional representation. The backgroup of the figure shows the density of songs in the *Map of Music*. If there are many songs in a region, the rectangle is bright.

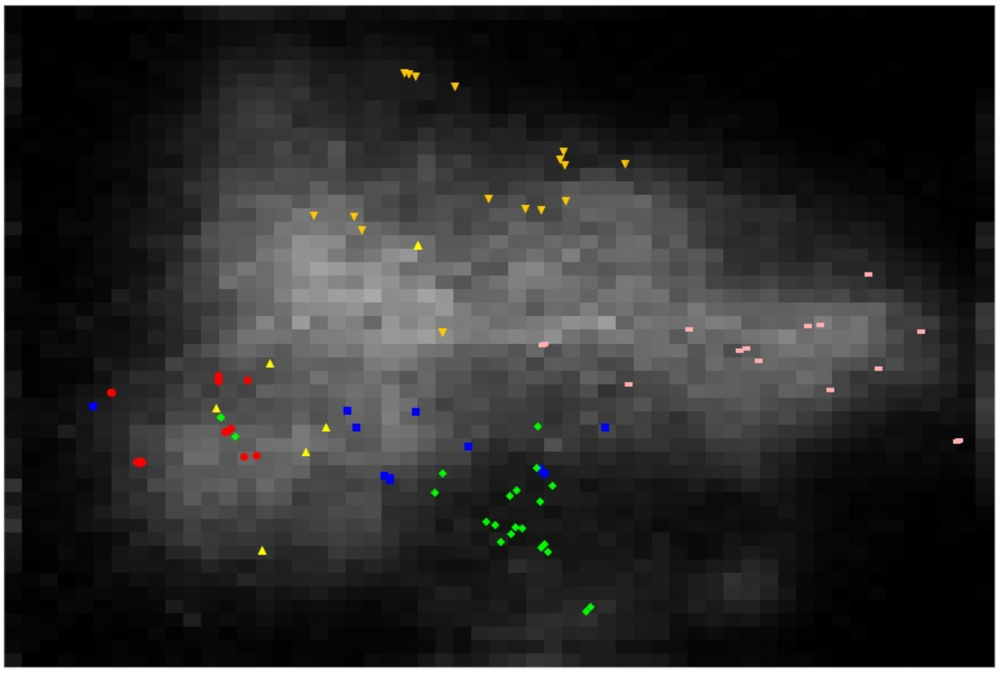


Figure 5.8.: Best clustering result for a set of songs

In Figure 5.9, the songs and path of the playlist received by the clustering method is shown. It is to remark, that not all songs presented in Figure 5.8 are part of the playlist, but instead there are songs which are part of the playlist, but not of the clustering. This arises from the smooth transition step at which additional songs of the embedding were added.

The result of the playlist generation with the adapted travelling salesman algorithm is displayed in Figure 5.10. Compared to the clustering method, there seem to be bigger hops and it looks more chaotic.

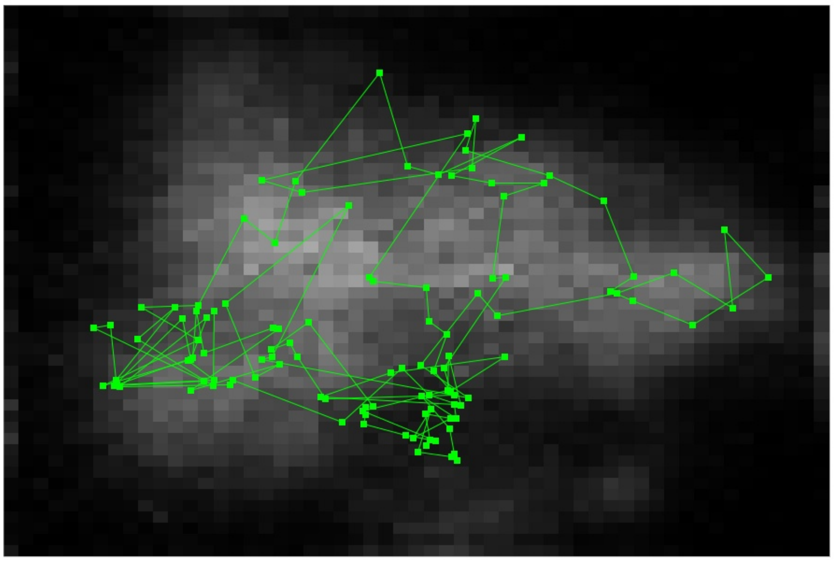


Figure 5.9.: Generated playlist for the clustering algorithm

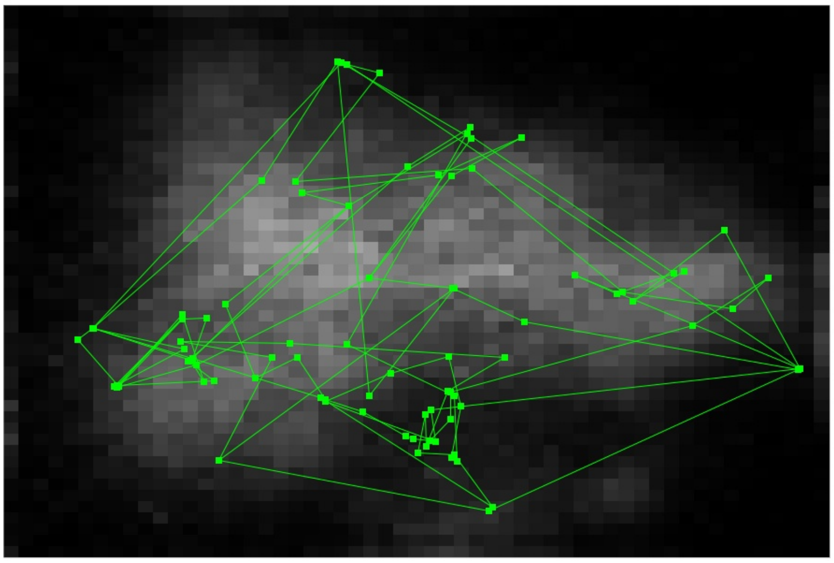


Figure 5.10.: Generated playlist for the salesman algorithm

6. DJ Player

Shortly before the party starts and during the party, only the DJ Player application is needed, which can be reached on the provided player website[13]. The application is a web application because it needs connection to the database with all the songs in the *Map of Music* and also to YouTube for playing videos.

6.1. Playing Songs

We built our DJ Player on top of the YouJuke Player[14]. The YouJuke Player consists of a client application, which is used by the party hosts, and an admin access, which is used for supervision and bug detection. In [20] the architecture of the YouJuke Player is described in detail.
We use YouTube as music source. A large number of songs in the embedding can be found on YouTube and additionally, we have the possibility to play videos. The YouTube API[15] has a wide range of functions. For our application, we use its ability to

- find songs and videos,
- filter bad quality videos,
- play them with the embedded YouTube player,
- and control the state of the player, such as playing the next song, when actual song is finished).

6.2. User Interface

6.2.1. Login system

When the organizer of the party opens the player website, he first has to log in to his party. In a listbox he can see all the upcoming parties and can choose his one and log in with the password defined while creating the party (see Section 4.5.2).

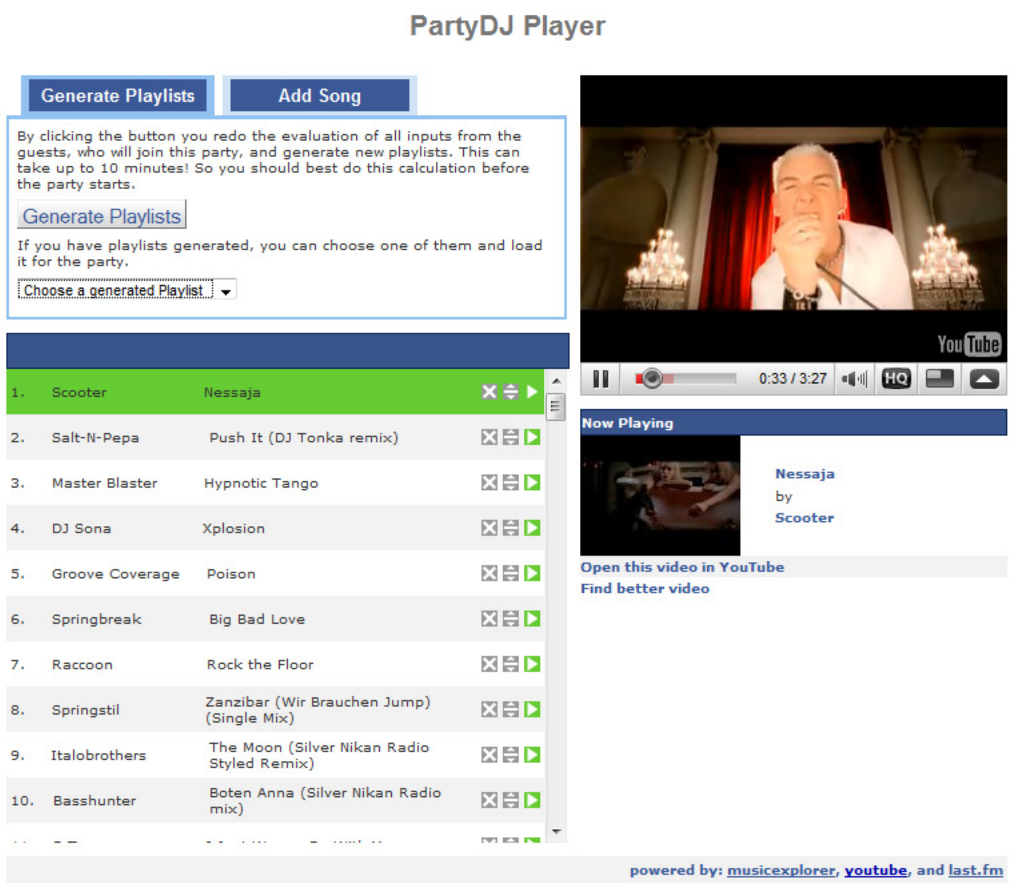
[13]http://pc-5413.ethz.ch/PartyDJPlayer/
[14]http://www.youjuke.org/YouJuke/YouJuke.html
[15]http://code.google.com/apis/youtube

6.2.2. Generate and Load Playlists

In the main window of the player the organizer can do (or redo) the playlist creation by clicking on the "Generate Playlists" button (see Figure 6.1). By doing this, the process of playlist generation is started.
Once the generation of the playlists is done, one of them can be chosen from the listbox below. There are five playlists which can be selected. The first one is created by using random songs in the *Map of Music*. The other were created by each combination of the two user procecessing methods and the two transistion methods, see Section 5.

6.2.3. The Playlist

Once a playlist is selected, the player appears as shown in Figure 6.1.



PartyDJ Player

Generate Playlists | Add Song

By clicking the button you redo the evaluation of all inputs from the guests, who will join this party, and generate new playlists. This can take up to 10 minutes! So you should best do this calculation before the party starts.

Generate Playlists

If you have playlists generated, you can choose one of them and load it for the party.

Choose a generated Playlist

1.	Scooter	Nessaja
2.	Salt-N-Pepa	Push It (DJ Tonka remix)
3.	Master Blaster	Hypnotic Tango
4.	DJ Sona	Xplosion
5.	Groove Coverage	Poison
6.	Springbreak	Big Bad Love
7.	Raccoon	Rock the Floor
8.	Springstil	Zanzibar (Wir Brauchen Jump) (Single Mix)
9.	Italobrothers	The Moon (Silver Nikan Radio Styled Remix)
10.	Basshunter	Boten Anna (Silver Nikan Radio mix)

0:33 / 3:27

Now Playing

Nessaja
by
Scooter

Open this video in YouTube
Find better video

powered by: musicexplorer, youtube, and last.fm

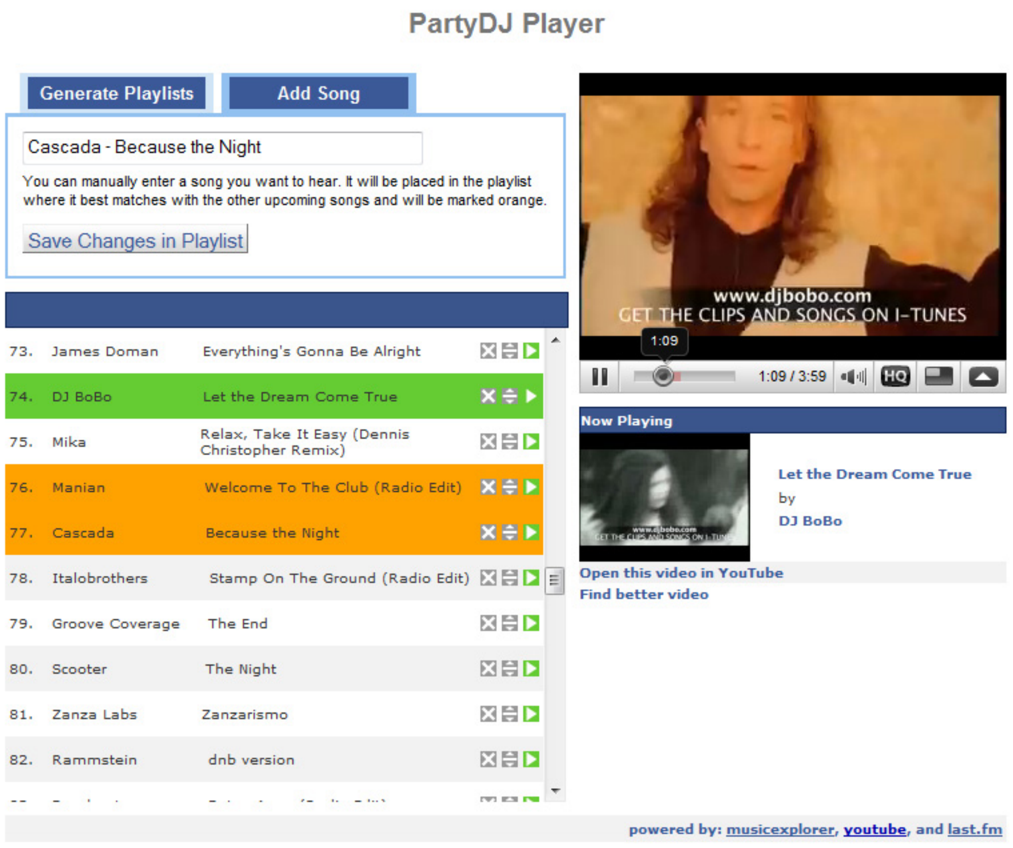
Figure 6.1.: PartyDJ player with loaded playlist

But the playlist does not have to be used as it is.The interface allows some modifica-

tions: On the right of each song there are symbols to delete or move the song with drag and drop and certainly to play it and songs can be added manually as described in the next section.
Actually playing songs are marked green.

6.2.4. Adding Songs

It is also possible to add songs manually during the party. Through the suggestion box, the partyguests can enter songs, which are present in the embedding and they will be placed where it best matches with the other songs. It uses the coordinates of the *Map of Music* to find the position in the playlist where the distance to the two adjoining songs is minimized. Manually inserted songs are marked orange, as shown in Figure 6.2.



Cascada - Because the Night

You can manually enter a song you want to hear. It will be placed in the playlist where it best matches with the other upcoming songs and will be marked orange.

73.	James Doman	Everything's Gonna Be Alright
74.	DJ BoBo	Let the Dream Come True
75.	Mika	Relax, Take It Easy (Dennis Christopher Remix)
76.	Manian	Welcome To The Club (Radio Edit)
77.	Cascada	Because the Night
78.	Italobrothers	Stamp On The Ground (Radio Edit)
79.	Groove Coverage	The End
80.	Scooter	The Night
81.	Zanza Labs	Zanzarismo
82.	Rammstein	dnb version

Let the Dream Come True
by
DJ BoBo

Open this video in YouTube
Find better video

powered by: musicexplorer, youtube, and last.fm

Figure 6.2.: PartyDJ player with manually added songs

All the changes in the playlist can be saved with the "Save Changes in Playlist" button.

6.2.5. YouTube Movies

For playing the songs, a YouTube player is integrated to the application. It plays videos for the songs, which can be found with the YouTube API. If the video is in bad quality or you just want to chose another one, a popup with alternative videos can be loaded by clicking at "Find better video".

7. User Studies

We have done two user studies to improve and evaluate the quality of our application. In an early stage of developing the PartyDJ application, we started the first user study with some of our friends on Facebook. The goal of this study was, that we receive some input from the users for better taking their preferences into account. With this aid it was easier to define some parameters in the processing users input step (Section 5.1) for the playlist generation.
In the final stage we have done a second study with members of the Distributed Computing Group at the ETH Zürich for a final evaluation of the PartyDJ application.

7.1. Questions

First, we asked the users in both studies some general questions about their age, taste of music and behaviour concerning Facebook and parties.
Form the user inputs, we created different playlists.

Playlists in the first study

1. **User Random Playlist:** This playlist is the simplest one, it just makes a random shuffling over all explicitly liked songs from the users who join the party
2. **User Weighted Playlist:** This playlist just makes a random shuffling over the best rated songs without using clustering or adapted nearest neighbourhood algorithm
3. **Smooth Playlist:** The main playlist which take into account all the calculated values for the songs and the clustering

Playlists in the second study

1. **Random Shuffeled Playlist:** This playlist is the simplest one, it just makes a random shuffling over randomly selected songs from the embedding
2. **Random Sorted Playlist:** This playlist uses randomly selected songs and sorts them with the adapted nearest neighbourhood algorithm (see Section 5.2.2)
3. **User Shuffeled Playlist:** This playlist is created using the songselection method based on artists as described in Section 5.1.2, but without any sorting

4. **User Sorted Playlist:** This playlist is created using the songselection method based on artists as described in Section

In the main part we asked always the same questions, once for every generated playlist. The users should imagine to be at a party and listen to the playlists and rate it. We asked questions, whose answers can be given in the likert scale [5]. This means, that they have the following possible answers (in brackets the value for mean and standard deviation calculation):

- Strongly disagree (-2)
- Disagree (-1)
- Neither agree nor disagree (0)
- Agree (+1)
- Strongly agree (+2)

Because all of the users (in the firs study) were from Switzerland or Germany, we have written the questions in German. We asked the following:

1. Ist der eigene Geschmack gut vertreten? (How well is your taste present?)
2. Bietet die Playlist einen guten Ablauf? (Has the playlist good transitions?)
3. Wie viel Abwechslung bietet die Playlist? (How much alternation is in the playlist?)
4. Wie stark regt die Playlist zum tanzen an? (How much does the playlist animate to dance?)
5. Gesamteindruck der Playlist (Overall impression of the playlist)

Only in the second study we also pleased the users to rate 11 songs from each of the four playlists.

7.2. Results

All detailed results from the studies can be found in appendix A. Here, we present the main.
Unfortunately most of the results in the first study are not in a statistical significant area, but it can clearly be seen, that the users' tastes are best present in the first playlist. The users also mostly prefered this playlist.
In the second study we have a bit more statistical relevance. Playlists 3 and 4 clearly better represent users' tastes (mean value: 0.3) than playlists 1 and 2 (mean value: -0.8). Also the 11 chosen songs from the playlists are better rated in playlists 3 and 4. Regarding the results from questions 2 and 5, the sorting seems not improve transition quality, but this result is not in a statistical significant area.

7.3. Conclusions

7.3.1. First study

The main conclusion from this study is, that users, who enter songs in the Facebook application and go to the party, primarily want to hear the songs, they have added. A smooth flow of the playlist does not have a high priority.
Because of this conclusion we gave a big "bonus" to all the songs that users added to their liked songs. In this way it is guaranteed, that this songs will appear in the playlist at the party. There is only one restriction about this: per user appears at most a certain fraction of the songs to ensure fairness between users. The fraction is proportional to the fraction a user takes from all users, e.g. if 20 users give inputs and the party has a duration of 6 hours, then 5 songs from his liked songs ar randomly taken (5 songs $\cong$ 18 min $\cong$ 5% of 360 min).

7.3.2. Second study

From the rating of the 11 songs per playlist can be seen, that the explicitly chosen songs from the users have a higher mean value than the other ones. Therefore, the decision, taken after the first study, to rate these songs higher and make sure that they are played is justified.
The second study showed that it is really a hard problem to satisfy all people, especially if the tastes of music diverge highly. With our approach we have at least a clear benefit against randomly chosen songs. Comparing the sorted playlists with the unsorted, there seems to be no improvent in user satisfaction. We question this result because it is hard to imagine to be at a party if there is no one. For a fair evaluation of playlist transition satisfaction, it would be needed to celebrate real parties.

8. Conclusion and Future Work

To realize the final goal of a "good" playlist, it is important to have knowledge about the music taste of the users, who will listen to the playlist. With this knowledge the song selection and song arranging can be done in different ways.
The task of collecting data about users taste of music was solved with an application for Facebook which also uses the the existing *Map of Music* and the social music community last.fm as source for songs, artists and genres. This application was used in two user studies and worked quite well.
In contrary to related approaches of playlist generation, which have the goal of generating playlists for a single user, we wanted to generate playlists for a group of people in the context of a private party. The evaluation of the two user studies showed, that this is really a tough problem, especially if we have a heterogeneous group of people where the tastes of music vary significantly between the users.
Figure 5.7 on page 26, which was created from the inputs of users who participated in one of the two studies, shows how difficult it is to find songs which most of the people would like. From the figure we can see, that certainly some songs in the region of "Hardstyle" and "Hands Up" should be played and songs in the Region of "Indie Pop" and "Blues" should be avoided. But what about other regions? It seems to be really impossible to find a playlist of songs which satisfies all people all the time and therefore it is not surprising that the average satisfaction of the users in our studies is not very high. The satisfaction is at least higher then for random playlists.
The evaluation of the first study showed, that for the users it is essential, that the songs, which they have personally entered, will really be played at the party. This is a fact which is also clear from the view of a DJ: if a person gets to the DJ at a party and wishes one specific song, then he really want to hear this song and not a song which seems to be quite similar. Finally when the DJ plays this song, the person is happy and his satisfaction is maximized for that moment.
It is also not really clear, how the satisfaction at a real party would be. We just generated the playlists with our algorithms and pleased the users to rate them. It would be an interesting topic to organise several parties with different created playlists for a better verification of our approaches.

A. Study Results

In the following we present the received results from our two user studies.

A.1. First study

A.1.1. User Information

In the first study with our friends, 20 people have rated songs, artists and genres with the Facebook application.
11 of them also gave their opinion about the playlists.

Mean age: 19.1
Gender: 10 male, 1 female

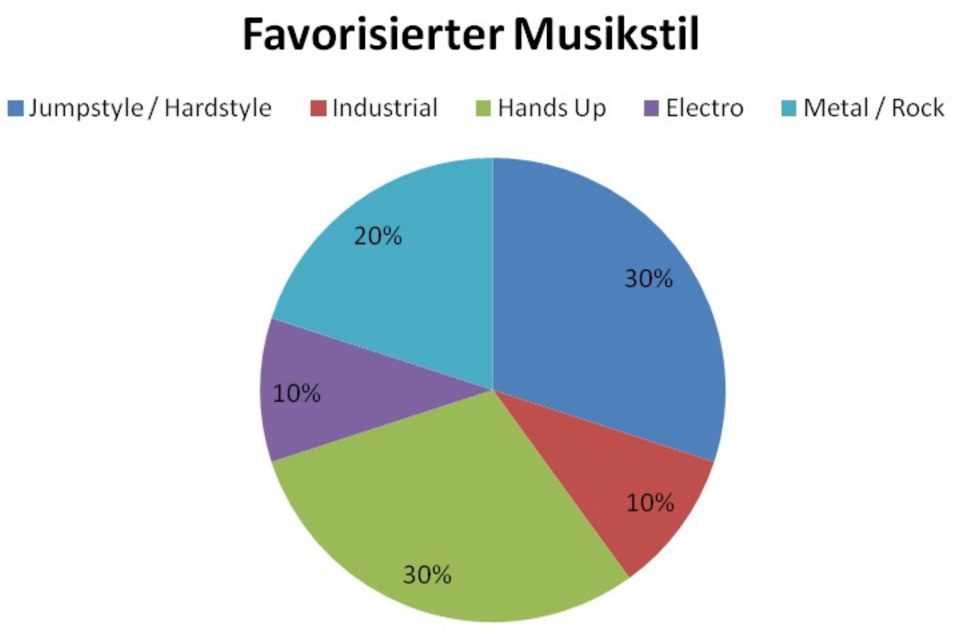


Figure A.1.: Favorite musicstyle of the users

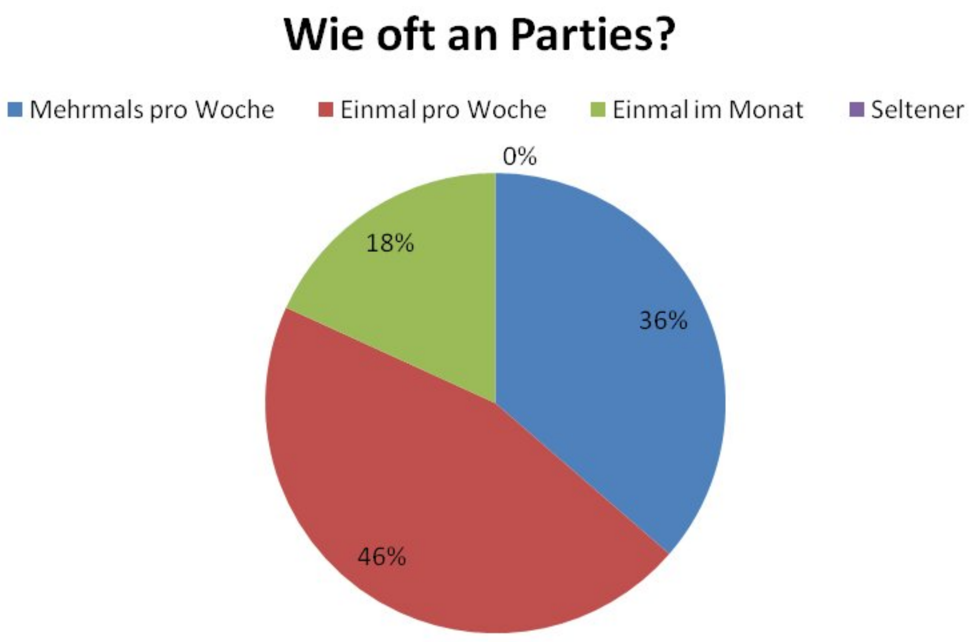


Figure A.2.: How often do they go to parties

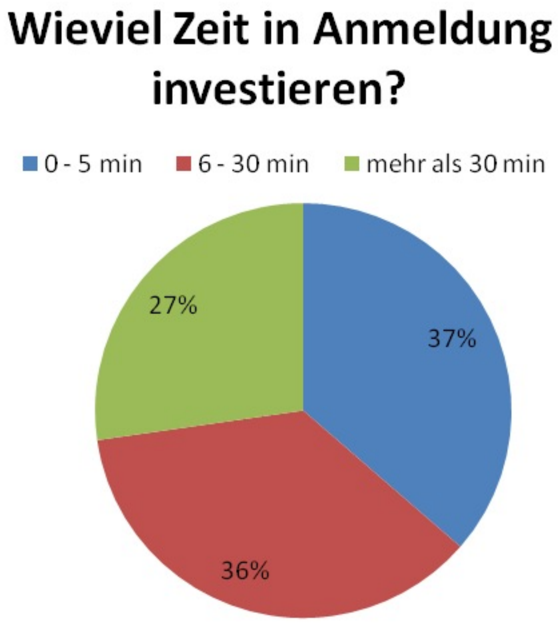


Figure A.3.: How much time would they give to rate entries in Facebook

A.1.2. Playlist Evaluation

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	1	0	1	5	4	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	1	2	4	3	1	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	0	0	8	2	1	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	0	1	2	5	3	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	0	1	4	4	2	Gesamteindruck gut

Table A.1.: Playlist 1 with random shuffled liked songs

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	1	1	4	3	2	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	0	2	3	4	2	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	0	1	8	2	0	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	0	4	3	3	1	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	0	3	4	2	2	Gesamteindruck gut

Table A.2.: Playlist 2 with random shuffled weighted songs

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	0	5	4	2	0	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	0	3	5	0	3	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	1	0	7	2	1	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	0	5	3	3	0	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	0	4	4	4	2	Gesamteindruck gut

Table A.3.: Playlist 3 generated according to clustering

Question	Playlist	mean	std.dev.
Ist der eigene Geschmack gut vertreten?	Playlist 1	1.00	1.13
	Playlist 2	0.36	1.15
	Playlist 3	-0.27	0.75
Bietet die Playlist einen guten Ablauf?	Playlist 1	0.09	1.08
	Playlist 2	0.55	0.99
	Playlist 3	0.27	1.14
Wie viel Abwechslung bietet die Playlist?	Playlist 1	0.36	0.64
	Playlist 2	0.09	0.51
	Playlist 3	0.18	0.94
Wie stark regt die Playlist zum tanzen an?	Playlist 1	0.91	0.90
	Playlist 2	0.09	1.00
	Playlist 3	-0.18	0.83
Gesamteindruck der Playlist	Playlist 1	0.64	0.88
	Playlist 2	0.27	1.05
	Playlist 3	0.09	0.67

Table A.4.: Mean values and standard deviations

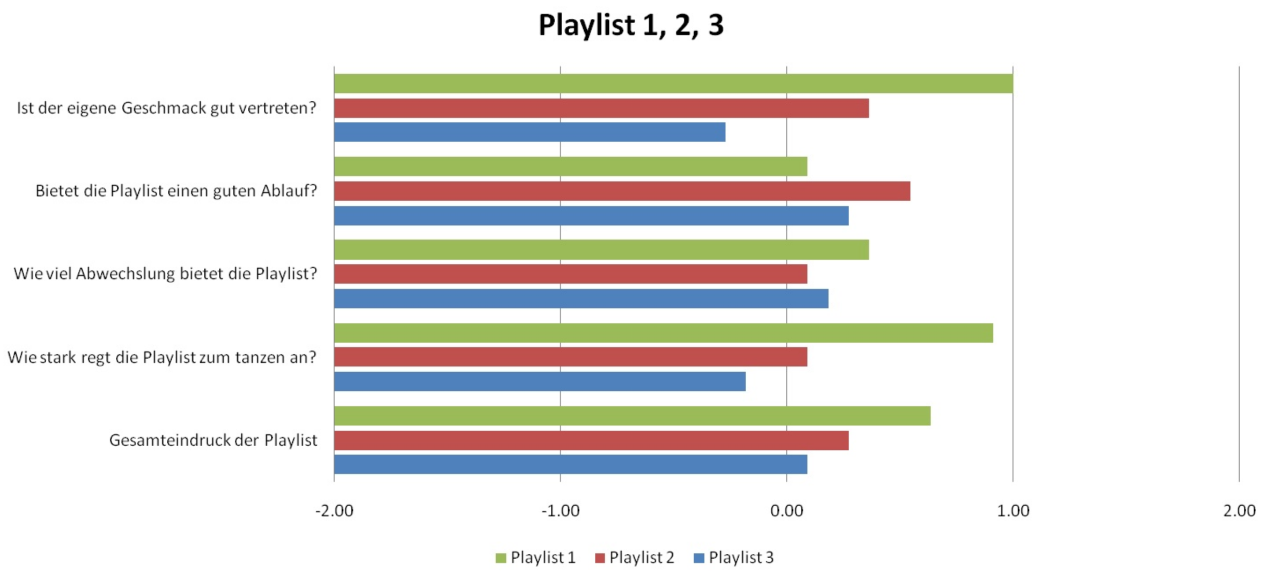


Figure A.4.: Comparison of the three playlists in the first study

A.2. Second study

A.2.1. User Information

In the second study with people of the Distributed Computing Group, 8 people have rated songs, artists and genres with the Facebook application.
5 of them also gave their opinion about the playlists.

Mean age: 30.6
Gender: 4 male, 1 female

A.2.2. Playlist Evaluation

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	2	1	2	0	0	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	2	2	0	1	0	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	0	0	3	0	2	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	3	1	1	0	0	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	3	1	1	0	0	Gesamteindruck gut

Table A.5.: Playlist 1 with random shuffled songs

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	1	2	1	1	0	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	1	2	1	1	0	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	0	1	1	1	2	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	3	0	1	1	0	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	1	2	2	0	0	Gesamteindruck gut

Table A.6.: Playlist 2 with random sorted songs

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	1	1	1	1	1	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	1	1	2	1	0	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	0	0	1	1	3	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	3	2	0	0	0	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	1	2	0	1	1	Gesamteindruck gut

Table A.7.: Playlist 3 with selected shuffled songs

	-2	-1	0	+1	+2	
Mein Geschmack ist schlecht vertreten	0	1	1	2	1	Mein Geschmack ist gut vertreten
Der Ablauf der Playlist ist schlecht	1	2	0	2	0	Der Ablauf der Playlist ist gut
Die Playlist hat zuwenig Abwechslung	0	0	1	2	2	Die Playlist hat zuviel Abwechslung
Bei dieser Playlist würde ich nicht tanzen	1	2	1	1	0	Bei dieser Playlist würde ich tanzen
Gesamteindruck schlecht	1	1	2	1	0	Gesamteindruck gut

Table A.8.: Playlist 4 with selected sorted songs

Question	Playlist	mean	std.dev.
Ist der eigene Geschmack gut vertreten?	Playlist 1	-1.00	0.89
	Playlist 2	-0.60	1.02
	Playlist 3	0.00	1.41
	Playlist 4	0.60	1.02
Bietet die Playlist einen guten Ablauf?	Playlist 1	-1.00	1.10
	Playlist 2	-0.60	1.02
	Playlist 3	-0.40	1.02
	Playlist 4	-0.40	1.20
Wie viel Abwechslung bietet die Playlist?	Playlist 1	0.80	0.98
	Playlist 2	0.80	1.17
	Playlist 3	1.40	0.80
	Playlist 4	1.20	0.75
Wie stark regt die Playlist zum tanzen an?	Playlist 1	-1.40	0.80
	Playlist 2	-1.00	1.26
	Playlist 3	-1.60	0.49
	Playlist 4	-0.60	1.02
Gesamteindruck der Playlist	Playlist 1	-1.40	0.80
	Playlist 2	-0.80	0.75
	Playlist 3	-0.20	1.47
	Playlist 4	-0.40	1.02
Bewertung von 11 Songs der Playlist	Playlist 1	-1.40	0.80
	Playlist 2	-0.80	0.75
	Playlist 3	-0.20	1.47
	Playlist 4	-0.40	1.02

Table A.9.: Mean values and standard deviations

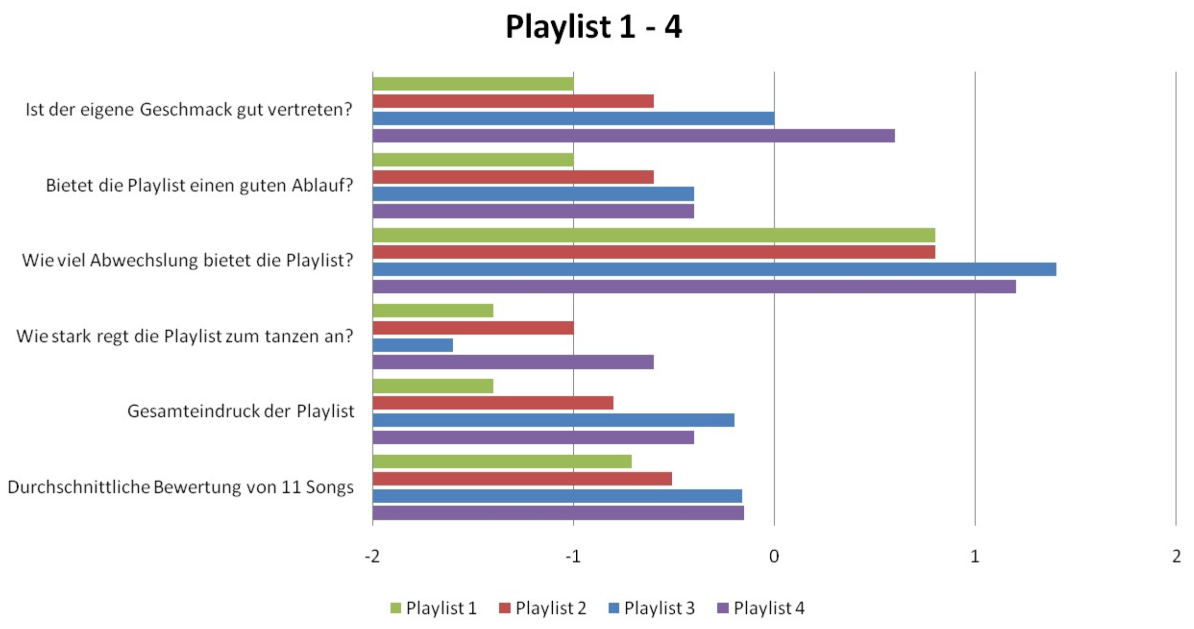


Figure A.5.: Comparison of the four playlists in the second study

B. List of Figures

C. List of Tables

D. Bibliography

[1] *Allmusic Genres*. Allmusic. – URL http://allmusic.com/cg/amg.dll?p=amg&sql=73:p. – Zugriff am 9. Dezember 2009

[2] *API - Last.fm*. Last.fm. – URL http://www.lastfm.de/api. – Zugriff am 9. Dezember 2009

[3] *Google Web Toolkit*. Google. – URL http://code.google.com/intl/de-CH/webtoolkit/. – Zugriff am 10. Dezember 2009

[4] *KD Tree*. Wikipedia. – URL Seehttp://en.wikipedia.org/wiki/Kd_tree. – Zugriff am 23. Dezember 2009

[5] *Likert scale*. Wikipedia. – URL http://en.wikipedia.org/wiki/Likert_scale. – Zugriff am 11. Dezember 2009

[6] *List of music styles*. Wikipedia. – URL http://en.wikipedia.org/wiki/List_of_music_styles. – Zugriff am 9. Dezember 2009

[7] *Pandora (music service)*. Wikipedia. – URL http://en.wikipedia.org/wiki/Pandora_%28music_service%29. – Zugriff am 23. Dezember 2009

[8] *Travelling Salesman Problem*. Wikipedia. – URL http://en.wikipedia.org/wiki/Travelling_salesman_problem. – Zugriff am 23. Dezember 2009

[9] *User:Google Web Toolkit*. Facebook Developers. – URL http://wiki.developers.facebook.com/index.php/User:Google_Web_Toolkit. – Zugriff am 10. Dezember 2009

[10] A. ANDRIC, G. H.: *Automatic playlist generation based on tracking user's listening habits*. Springer Science + Business Media. May 2006

[11] A. CROSSEN, K. H.: *Flytrap: Intelligent Group Music Recommender*. IUI'02, San Francisco, California, USA. 2002

[12] A. FLEXER, M. Gasser G. W.: *Playlist Generation Using Start and End Song*. ISMIR 2008. 2008

[13] A. JAMESON, B. S.: *Recommendation to Groups (in: The Adaptive Web)*. Springer-Verlag, Berlin, Germany. 2007

[14] FUNG, G.: *A Comprehensive Overview of Basic Clustering Algorithms*. 2001

[15] G. GUTIN, A. Z.: *Traveling salesman should not be greedy*. 2001

[16] J. AUCOUTURIER, F. P.: *Scaling Up Music Playlist Generation*. Sony Computer Science Laoratory, Paris, France. 2007

[17] J. FRENCH, D. H.: *Flycasting: On the Fly Broadcasting*. DELOS Network of Excellence Workshop, Dublin City, Ireland. 2001

[18] L. BOSSARD, R. W.: *Visually and Acoustically Exploring the High-Dimensional Space of Music*. IEEE International Conference on Social Computing (SocialCom), Vancouver, Canada. August 2009

[19] LLOYD, S.: *Least squares quantization in pcm*. 1982

[20] M. CALIN, M. Kuhn R. W.: *Jukebox - An Intelligent Online Music Player*. ETH Zürich. April 2009

[21] M. HALKIDI, M. V.: *Cluster validity methods: part I*. 2002

[22] M. HALKIDI, M. V.: *Cluster validity methods: part II*. 2002

[23] M. LORENZI, M. Kuhn R. W.: *FromWeb to Map: Exploring the World of Music*. IEEE/WIC/ACM International Conference on Web Intelligence (WI), Sydney, Australia. December 2008

[24] O. GOUSSEVSKAIA, R. W.: *Exploring Music Collections on Mobile Devices*. International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI), Amsterdam, Netherlands. September 2008

[25] S. PAUWS, B. E.: *PATS: Realization and User Evaluation of an Automatic Playlist Generator*. IRCAM, Centre Pompidou, Paris, France. 2002