An Implementation of a Simple Playlist Generator Based on Audio Similarity Measures and User Feedback

Elias Pampalk*
National Institute of Advanced
Industrial Science and Technology (AIST)
IT, AIST, 1-1-1 Umezono
Tsukuba, Ibaraki 305-8568, Japan

Martin Gasser
Austrian Research Insitute
for Artificial Intelligence (OFAI)
Freyung 6/6
A-1010 Vienna, Austria

Abstract

This paper presents an implementation of a simple playlist generator. An audio-based music similarity measure and simple heuristics are used to create playlists given minimum user input. The ultimate goal of this work is to conduct a field study, i.e., to run the system on the users' personal collection and study the usage behavior over a longer period of time. The functions include, for example, allowing the user to control the variance of the playlists in terms of how often the same song or songs from the same artists are repeated.

1. Introduction

Mobile audio players can store personal music collections of 20,000 and more tracks. However, the value of such large collections is limited by how easy it is, for example, to create an interesting playlist. In the MIR research community several approaches have been presented addressing this (see e.g. [1, 2, 3, 4, 5]).

A number of commercial systems exist which create interesting playlists. For example, iTunes allows the user to create "smart playlists" which use simple rules based on metadata and song ratings. Last.fm uses collaborative filtering to create playlists of similar songs and pandora.com uses manually annotated data. In contrast to these systems our implementation does not require any additional information other than the audio signals. In particular, it uses computational models of audio-based music similarity.

In this paper we present an implementation of a simple playlist generator (SPG) based on the work presented in [6]. The idea is to use simple heuristics to constantly improve playlists by adjusting them to user feedback. The minimum interaction scenario is that the user selects one song and presses the play button. SPG would respond by playing similar songs. If the user rates one or more of these songs this information will be used to improve future recommendations.

Our implementation serves two purposes. First, it has helped clarify the requirements of a simple playlist generator. Including, for example, the requirements of adjusting the variance in the playlists mentioned below. Second, it allows us to conduct user studies where the users can use the tool on their own music at home. In particular, our goal is to ask the users to keep daily notes of their experiences, and follow up these descriptions with structured interviews.

2. Functionality

SPG uses an audio-based similarity measure, feedback provided by the user, and simple heuristics. These heuristics search for songs similar to preferred songs and avoid songs similar to rejected songs. The usage scenario is that the users want to quickly create a playlist from their own music collection.

The core functions of SPG are:

A. Very similar to pandora.com the users can manage their own radio stations. (However, in this case a radio station only plays songs from the user's collection.) Each radio station is defined by favorite and banned songs or artists. Depending on the ratings the users make they can define a radiostation to play what they consider to be, for example, "wake-up music" or "Saturday day night music". The minimum requirement to define a new radio station is to select one favorite song or artist. All the user's ratings can be easily accessed and modified any time.

B. A key issue is to control the variance of the songs played on the radio station. If there is too little variance the same songs will be repeated frequently, if there is too much variance the songs on the playlist might not be similar to each other. The user can control the variance by setting the frequency with which songs and artists are repeated. For each there are three options: rarely, sometimes, and frequently.

C. In contrast to the work presented in [6] skipped songs, and songs which are played, are not automatically rated. Instead the user can easily adjust the rating of each song in the playlist or in the edit radio station panel.

*) Part of this work was done while the author was working at the Austrian Research Institute for Artificial Intelligence (OFAI).

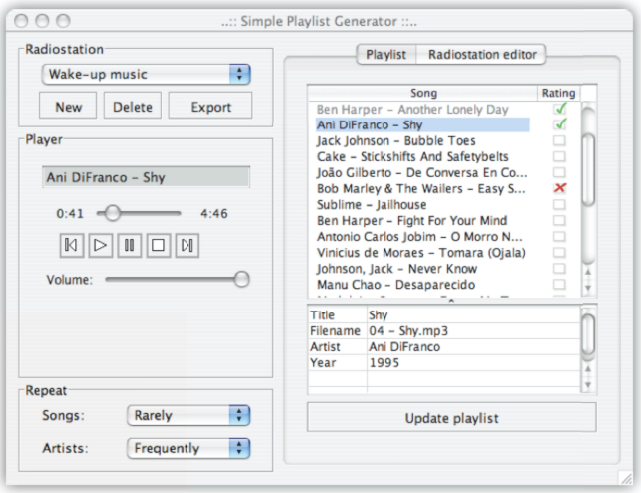

Figure 1. Screenshot of SPG running on MacOS X.

3. User Interface

Figure 1 shows a screenshot of SPG (which is implemented in Java). The list on the right side is the current playlist which is generated by the selected radio station. The user can rate songs by clicking on the icons to the right of each song. (This only has an impact on the current radio station.) The playlist is updated by clicking on "update playlist". On the left side (top to bottom) are the radio station selector, audio player controls, and controls to adjust the variance of the playlist. The export button underneath the playlist selector allows the users to export the current playlist to an M3U file (and thus allows them to use their preferred audio player).

The controls to adjust the variance allow the user to set the number of times artists and songs are repeated to either "rarely", "sometimes", or "frequently". If, for example, the user chooses to repeat songs rarely then SPG would rather play a not so similar song instead of playing a previously played song again. If the user chooses to repeat artists frequently then SPG (if appropiate) might play a number of songs by the same artist within an hour. The user can only set the repetition frequency of artists to the maximum level the user chose for the songs (e.g. repeating songs frequently, but artists rarely is not a valid setting).

Not shown is the edit radio station panel where the users can view their ratings and change the name of the radio station. One list is shown for all rated songs, and one list for all rated artists. New items can be added using a simple search function. Ratings in the lists can be changed in the same way that they are changed in the playlist.

4. Techniques

We use the audio-based music similarity measure described as G1C in [7]. It combines spectral similarity [8] with information from fluctuation patterns [9, 10]. To create playlists we use the heuristic D described in [6]. Basically, songs which are close to any of the user's favorite songs, and far away from banned songs are recommended. If the user dislikes or likes an artist then all songs from this artist (unless they are rated individually) are treated as favorite or banned songs.

5. Conclusions

In this paper we presented a Java implementation of a simple playlist generator. The player implements minimum functionality to support evaluation of MIR technologies (in particular audio-based music similarity measures and playlist generation heuristics) in everyday music consumption. The functionality includes management of radio stations and simple control of the variance in the play-lists. Future work includes conducting a user study where users install the tool on their private collections.

6. Acknowledgments

This work was supported by the EU project FP6-507142 (SIMAC) and the WWTF project Interfaces to Music (I2M).

References

[1] J.-J. Aucouturier and F. Pachet, "Scaling up music playlist generation," in *Proc. of the IEEE Intl. Conf. on Multimedia Expo*, 2002.

[2] S. Pauws and B. Eggen, "PATS: Realization and User Evaluation of an Automatic Playlist Generator," in *Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, 2002.

[3] T. Pohle, E. Pampalk, and G. Widmer, "Generating Similarity-Based Playlists Using Traveling Salesman Algorithms," in *Proc. of the Intl. Conf. on Digital Audio Effects*, 2005.

[4] R. van Gulik and F. Vignoli, "Visual Playlist Generation on the Artist Map," in *Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, 2005.

[5] M. Goto and T. Goto, "Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces," in *Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, 2005.

[6] E. Pampalk, T. Pohle, and G. Widmer, "Dynamic Playlist Generation Based on Skipping Behaviour," in *Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, 2005.

[7] E. Pampalk, "Computational Models of Music Similarity and their Application in Music Information Retrieval," Ph.D. dissertation, Vienna University of Technology, 2006.

[8] M. Mandel and D. Ellis, "Song-Level Features and Support Vector Machines for Music Classification," in *Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, 2005.

[9] E. Pampalk, "Islands of Music: Analysis, Organization, and Visualization of Music Archives," Master's thesis, Vienna University of Technology, 2001.

[10] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of Audio-Based Music Similarity and Genre Classification," in *Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, 2005.