DYNAMIC PLAYLIST GENERATION BASED ON SKIPPING BEHAVIOR

Elias Pampalk[1], **Tim Pohle**[1], **Gerhard Widmer**[1,2]
[1]Austrian Research Institute for Artificial Intelligence (OFAI), Freyung 6/6, 1010 Vienna, Austria
[2] Department of Computational Perception, Johannes Kepler University, Linz, Austria

ABSTRACT

Common approaches to creating playlists are to randomly shuffle a collection (e.g. iPod shuffle) or manually select songs. In this paper we present and evaluate heuristics to adapt playlists automatically given a song to start with (seed song) and immediate user feedback.

Instead of rich metadata we use audio-based similarity. The user gives feedback by pressing a skip button if the user dislikes the current song. Songs similar to skipped songs are removed, while songs similar to accepted ones are added to the playlist. We evaluate the heuristics with hypothetical use cases. For each use case we assume a specific user behavior (e.g. the user always skips songs by a particular artist). Our results show that using audio similarity and simple heuristics it is possible to drastically reduce the number of necessary skips.

1 INTRODUCTION

There are different ways to create playlists. One extreme is to very carefully select each piece and the order in which the pieces are played. Another extreme is to randomly shuffle all pieces in a collection. While the first approach is very time consuming, the second approach produces useless results if the collection is very diverse.

In this paper we present an alternative which requires little user interaction even for very diverse collections. The goal is to minimize user input and maximize satisfaction. Our *assumptions* are (1) a seed song is given. We do not tackle the problem of browsing a large collection to find a song to start with. If more than one song to start with is given then the task is simplified. (2) We assume that a skip button is available and easily accessible to the user. For example, this is the case if the user runs Winamp while browsing the Internet. (3) Finally, we assume a lazy user who is willing to sacrifice quality for time. In particular, we assume all the user is willing to do is press a skip button if the song currently played is a bad choice.

In this paper we present simple heuristics to dynamically propose the next song to be played in a playlist. The approach is based on audio similarity and we take the user's skipping behavior into account. The idea is to avoid songs similar to songs which were skipped and focus on songs similar to accepted ones. We evaluate the heuristics based on hypothetical use cases.

Previous work on playlist generation has partly dealt with algorithms to efficiently find a playlist which fulfills given constraints (e.g. [1, 2]). These approaches assume the existence of rich metadata. A commercial product to generate playlists from proprietary metadata is available from Gracenote.[1] This "dynamic" playlist generator updates playlists when the music collection is modified.

A conceptually very similar approach to our work is the internet radio station Last.FM[2] which creates a user profile based on immediate user feedback. Last.FM is built on collaborative filtering and uses "Love", "Skip", "Ban" buttons as input.

Another approach using immediate user feedback and rich metadata was presented in [3]. The main difference to our work is that in our evaluations random shuffling would completely fail. Furthermore, our heuristics have no parameters which need to be trained and thus we expect our approach to be more robust and require less user feedback.

Unlike these previous approaches we do not rely on metadata or collaborative filtering. Purely audio-based playlist generation was proposed for example in [5] and [4]. In [5] the author showed that simply using the n nearest songs to a seed song as a playlist performs relatively well. In [4] traveling salesman algorithms are used to find a path through the whole collection.

2 METHOD

At the core of our approach is the audio-based music similarity measure. We use a combination of spectral similarity [6, 7], fluctuation patterns [8], and some other descriptors. The details are described in [9].

From a statistical (or machine learning) point of view it is problematic to use complex models (or learners) with a high number of parameters to learn user preferences. The main reason for this is that in an ideal case the number of negative examples is extremely low (e.g. less than 5) and even the number of positive examples is not very high (e.g. 20 tracks can fill an hour of music).

To generate the playlists we use the following 4 *heuristics*. Candidate songs are all songs in the collection which have not been played (or skipped) yet.

(**A**) As suggested in [5] the n nearest neighbors to the seed song are played (n = accepted + skipped). This heuristic creates a static playlist and is the baseline we want to improve upon.

(**B**) The candidate song closest to the last song accepted by the user is played. This is similar to heuristic A with the only difference that the seed song is always the last song accepted.

(**C**) The candidate song closest to any of the accepted songs is played. Using the minimum distance for recom-

[1]http://www.gracenote.com/gn_products/playlist.html
[2]http://last.fm

mendations from song sets was proposed in [10].

(**D**) For each candidate song, let d_a be the distance to the nearest accepted, and let d_s be the distance to the nearest skipped. If $d_a < d_s$, then add the candidate to the set S. From S play the song with smallest d_a. If S is empty, then play the candidate song which has the best (i.e. the lowest) d_a/d_s ratio.

3 EVALUATION

In the hypothetical use cases we assume that the user wants to listen to one hour of music which is approximately 20 songs. The number of skips are counted until these 20 songs are played. The *use cases* (UC) are the following:

(**1**) The user wants to listen to songs similar to the seed. We measure this by equating similarity with genre membership. Any song outside of the seed's genre is skipped.

(**2**) The user wants to listen to similar music but dislikes a particular artist (for not measurable reasons such as personal taste). To measure this we use the same approach as for UC-1. An unwanted artist from the seed's genre (not the artist of the seed song) is randomly selected. Every time a song outside the seed's genre or from the unwanted artist is played, skip is pressed.

(**3**) The user's preferences change over time. We measure this as follows. Let A be the genre of the seed song and B a related genre which the user starts to prefer. The first 5 songs are accepted if they are from genre A. The next 10 are accepted if they are from either A or B. The last 5 are accepted if they are from B. We manually selected pairs of genres for this use case. The list of pairs can be found in Table 3. Unlike UC-1 and UC-2 it is possible that in UC-3 a state is reached where none of the candidate songs would be accepted although the number of accepted is less than 20. In such cases the remaining songs in the collection are added to the skip count.

For UC-1 and UC-2 the evaluation is run using every song in the collection as seed. For UC-3 every song in genre A is used.

One of the biggest problems for our evaluation is that we do not have enough artists per genre to implement an artist filter. That is, we do not avoid playing several songs from the same artist right after each other.

Another issue is that we assume only songs the user dislikes are skipped. However, if a song is skipped because, e.g., the user just heard it on the radio (but likes it otherwise) our heuristics will be mislead. To evaluate this we could have included randomly pressed skips. To solve this the user could be given more feedback options. For example, how long or hard the skip button is pressed could indicate how dissimilar the next song should be.

3.1 Data

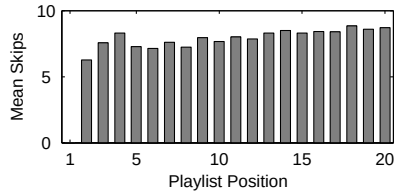
The collection we use contains 2522 tracks from 22 genres (see Table 1 for further statistics). The genres and the number of tracks per genre are listed in Fig. 2. The collection has mainly been organized according to genre/artist/album. Thus, all pieces of an artist are assigned to the same genre, which is questionable but common practice. The genres are user defined and inconsistent. In particular, there are two different definitions of trance. Furthermore, there are overlaps, for example, jazz and jazz guitar, heavy metal and death metal etc.

			Artists/Genre		Tracks/Genre	
Genres	Artists	Tracks	Min	Max	Min	Max
22	103	2522	3	6	45	259

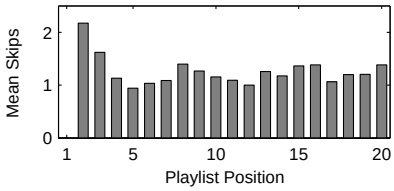
Table 1: Statistics of the music collection.

	Heuristic	Min	Median	Mean	Max
UC-1	A	0	37.0	133.0	2053
	B	0	30.0	164.4	2152
	C	0	14.0	91.0	1298
	D	0	11.0	23.9	425
UC-2	A	0	52.0	174.0	2230
	B	0	36.0	241.1	2502
	C	0	17.0	116.9	1661
	D	0	15.0	32.9	453

Table 2: Number of skips for UC-1 and UC-2.



(a) Heuristic A



(b) Heuristic D

Figure 1: Skips per playlist position for UC-1.

3.2 Results

For UC-1 using random shuffle to generate the playlist would require more than 300 skips in half of the cases while heuristic A requires less than 37 skips in half of the cases. Table 2 shows the results for UC-1 and UC-2. The main observation is that the performance increases from heuristic A to D. In general, there are a lot of outliers which is reflected in the large difference between mean and median. In a few cases almost all songs from the collection are proposed until 20 songs from the seed genre are in the playlist. Heuristic D has significantly fewer outliers. Half of all cases for heuristic D in UC-1 require less than 11 skips which might almost be acceptable.

Fig. 1 shows that for D/UC-1 there is a large number of skips after the first song (seed song). Once the system has a few positive examples the number of skips decreases. On the other hand, for heuristic A, the number of skips gradually increases with the playlist position. (Note that one must be careful when interpreting the mean because it is strongly influenced by a few outliers.)

Fig. 2 shows that for D/UC-1 some genres work very well (e.g. jazz guitar or heavy metal - trash), while others

Start	Goto	Heuristic A Median	Heuristic A Mean	Heuristic B Median	Heuristic B Mean	Heuristic C Median	Heuristic C Mean	Heuristic D Median	Heuristic D Mean
Euro-Dance	Trance	69.0	171.4	36.0	64.9	41.0	69.0	20.0	28.3
Trance	Euro-Dance	66.0	149.1	24.0	79.1	6.5	44.4	4.5	8.8
German Hip Hop	Hard Core Rap	33.0	61.9	32.0	45.6	31.0	40.7	23.0	28.1
Hard Core Rap	German Hip Hop	21.5	32.2	18.0	51.9	16.0	24.2	14.0	16.1
Heavy Metal/Thrash	Death Metal	98.5	146.4	54.0	92.5	58.0	61.1	28.0	28.4
Death Metal	Heavy Metal/Thrash	14.0	69.2	16.0	53.7	3.0	55.5	3.0	25.7
Bossa Nova	Jazz Guitar	68.5	228.1	32.0	118.7	54.0	61.1	22.0	21.3
Jazz Guitar	Bossa Nova	21.0	26.7	22.0	21.5	9.0	10.5	6.0	6.2
Jazz Guitar	Jazz	116.0	111.3	53.0	75.7	45.0	74.0	18.5	27.3
Jazz	Jazz Guitar	512.5	717.0	1286.0	1279.5	311.0	310.8	29.0	41.3
A Cappella	Death Metal	1235.0	1230.5	1523.0	1509.9	684.0	676.5	271.0	297
Death Metal	A Cappella	1688.0	1647.2	1696.0	1653.9	1186.0	1187.3	350.0	309.2

Table 3: Number of skips for UC-3.

fail (e.g. electronic or downtempo). However, some of the failures make sense. For example, before 20 pieces from electronic are played, in average almost 18 pieces from downtempo are proposed.

Table 3 gives the results for UC-3. As for the other use cases the performance increases from A to D in most cases. We have included the pair a capella to death metal as an extreme to show the limitations (we do not consider such a transition to be a likely user scenario). In three of the cases for heuristic D the median seems to be acceptably low.

The number of skips depends a lot on the direction of the transition. For example, moving from jazz guitar to bossa nova requires, in half of the cases, less than 6 skips. Moving in the other direction requires almost 3 times as many skips. This is also reflected in Fig. 2. Specifically, jazz guitar to bossa nova works well because jazz guitar is mainly confused with bossa nova. On the other hand bossa nova is confused with many other genres. The same can be observed, e.g., for the pair trance and euro-dance.

Fig. 3 shows where skips occur for UC-3 and heuristic D, and how often each genre was played per playlist position. In some cases during the transition phase (where genre A or B are accepted) basically only genre A is played. When the transition is forced (after the 15th song in the playlist) the number of skips drastically increases. In other cases the transition works very nicely. An obvious direction for further improvement is to include a memory effect to allow the system to quickly forget previous user choices. However, preliminary experiments we conducted in this direction did not show significant improvements.

4 CONCLUSIONS

We have presented an approach to dynamically create playlists based on the user's skipping behavior. We evaluated the approach using hypothetical use cases for which we assume specific behavior patterns. Compared to the approach suggested in [5], heuristic D reduces the number of skips drastically. In some of the cases the necessary number of skips seems low enough for a real world application.

The main limitation of our evaluation is that we did not implement an artist filter (to avoid having a large number of pieces from the same artist right after each other in a playlist) due to the small number of artist per genre.

The heuristic depends most of all on the similarity measure. Any improvements would lead to fewer skips. However, implementing memory effects (to forget past decisions of the user) or allowing the similarity measure to adapt to the user's behavior are also interesting directions. For use cases related to changing user preferences a key issue might be to track the direction of this change. Incorporating additional information such as web-based artist similarity or modeling the user's context more accurately (based on data from long term usage) are other options.

Although evaluations based on hypothetical use cases seems to be sufficient for the current development state, experiments with humans listeners will be necessary in the long run.

Acknowledgements

This research was supported by the EU project FP6-507142 (SIMAC). OFAI is supported by the Austrian ministries BMBWK and BMVIT.

References

[1] M. Alghoniemy and A. Tewfik. A network flow model for playlist generation. In *Proc IEEE Intl Conf Multimedia and Expo*, 2001.

[2] J.-J. Aucouturier and F. Pachet. Scaling up music playlist generation. In *Proc IEEE Intl Conf on Multimedia Expo*, 2002.

[3] S. Pauws and B. Eggen. PATS: Realization and user evaluation of an automatic playlist generator. In *ISMIR*, 2002.

[4] T. Pohle, E. Pampalk, and G. Widmer. Generating similarity-based playlists using traveling salesman algorithms. In *Proc Intl Conf Digitial Audio Effects*, 2005.

[5] B. Logan. Content-based playlist generation: Exploratory experiments. In *Proc ISMIR*, 2002.

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

[7] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proc IEEE Intl Conf on Multimedia and Expo*, 2001.

[8] E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. MSc thesis, Vienna University of Technology, 2001.

[9] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proc ISMIR*, 2005.

[10] B. Logan. Music recommendation from song sets. In *Proc ISMIR*, 2004.

		AC	AJ	Blu	BN	Cel	DM	DnB	DT	Ele	ED	FR	GH	HC	HM	Ita	Jaz	JG	MM	Pun	Roc	Tra	T2	Sum
A Cappella	112		0.3	0.1	0.4	0.2		0.2	0.6	0.3	0.2	0.5	0.4	0.2	0.2	0.5	0.1	0.1	0.2	0.2	0.1	0.3	0.1	4.9
Acid Jazz	68	1.2		2.4	1.0	2.9	0.4	5.0	7.8	3.6	4.1	7.5	8.9	5.5	0.6	6.1	4.3	1.1	0.8	3.4	1.9	3.5	2.5	74.4
Blues	63	0.1	2.3		0.9	2.9	0.3	0.2	0.7	0.9	0.1	3.6	1.1	1.0	0.5	13.9	2.7		5.4	4.0	1.2	0.1	0.1	42.0
Bossa Nova	72	0.3	0.1	1.8		2.4				1.7	0.3	0.5	2.5		1.0	4.6	3.0	1.4					0.8	20.5
Celtic	132	0.8		1.6	0.5		0.7			0.2	0.1	1.1	0.1	0.3	0.3	3.9	0.2	0.2	1.0	1.3				12.5
Death Metal	72	0.3	0.2	0.3	0.2	0.9		0.8	1.2	0.9	0.3	5.3	0.6	0.3	5.2	0.7	0.2	0.2	2.5	7.5	0.1	0.6	0.2	28.3
DnB	72	0.3	1.2	0.2	0.1	0.1			1.0	0.3	2.7	3.2	3.1	10.0	0.7	1.7	1.8		0.1	1.9	2.6	1.2	0.8	32.9
Downtempo	124	1.6	4.2	2.9	2.5	3.7	0.6	4.0		5.8	4.0	6.2	8.6	4.7	2.8	3.3	3.0	1.7	2.2	4.7	1.8	3.1	2.8	74.4
Electronic	63	4.0	8.5	1.6	5.0	7.9	2.3	13.0	17.9		10.2	8.5	8.5	6.3	2.5	4.3	3.6	8.5	1.9	3.1	2.0	8.0	4.2	131.8
Euro−Dance	97	0.1	0.2	0.1	0.1	0.3	0.1	0.3	0.5	0.4		0.3	0.6	0.5	0.1	0.1	0.1		0.1	0.1	0.1	0.7	0.6	5.5
Folk−Rock	238	0.5	0.1	0.4	0.3	0.4	0.9	0.1	0.2	0.2	0.2		0.6	0.5	2.6	1.8	0.2	0.1	2.4	4.2	0.4	0.1	0.1	16.1
German Hip Hop	143	0.7	0.8	0.8	0.4	0.7		0.3	0.7	0.3	0.2	1.3		1.9	0.2	4.1	1.9	0.1	0.2	1.0	0.5	0.2	0.1	16.3
Hard Core Rap	164	0.4	1.6	1.2		0.5		0.8	0.2	0.3		3.2	8.3		0.1	4.9	4.0		0.2	2.7	3.2	0.1		31.9
Heavy Metal − Thrash	242					0.1	0.2		0.1	0.1		0.3	0.1			0.2	0.1		0.1	1.4				2.7
Italian	142	0.1	0.1	0.1	0.2	0.2	0.1	0.3	0.4	0.2	0.1	0.5	0.5	0.5	0.3		0.1	0.1	0.2	1.3	0.1	0.2		5.7
Jazz	64	0.5	5.9	3.6	4.4	2.6	0.1	0.3	3.3	0.3	0.6	2.5	12.3	7.6	0.7	9.2		2.2	1.2	1.8	1.7	0.2	0.2	61.1
Jazz Guitar	70				0.7												0.3							1.1
Melodic Metal	169	0.1	0.1	0.2	0.1	0.8	0.9	0.2	0.3	0.2	0.2	3.3	0.2	0.1	1.1	0.5	0.1			5.2		0.1	0.1	13.6
Punk	259	0.1	0.1			0.1	0.5	0.1	0.2	0.1		0.3	0.1	0.1	0.1	0.2	0.1		0.1			0.1		2.3
Reggae	47	1.2	1.0	1.7	0.3	1.5		0.8	0.7	0.9	0.7	1.1	5.0	10.2	0.2	3.7	1.2	0.6	0.1	0.6		0.2	0.2	32.1
Trance	64	0.7	2.6		0.5	1.3	0.9	2.3	2.5	2.9	16.1	4.1	3.1	1.4	1.0	2.3	0.6	0.7	2.0	4.0	0.4		4.2	53.6
Trance2	45	0.6	2.7	0.4	0.5	0.6	0.1	1.7	3.4	2.5	15.2	2.5	1.5	1.6	1.1	0.4	0.4	0.4	0.9	1.1	1.2	2.7		41.5

Figure 2: Mean number of skips per genre for heuristic D in UC-1. For example, the first line shows how many songs (in average, computed as the mean) from each genre were skipped for playlists starting with an a capella seed. The number to the left of the table (e.g. 112) is the number of total tracks in the genre. The number on the right side of the table (4.9) is the mean of the total number of skips.

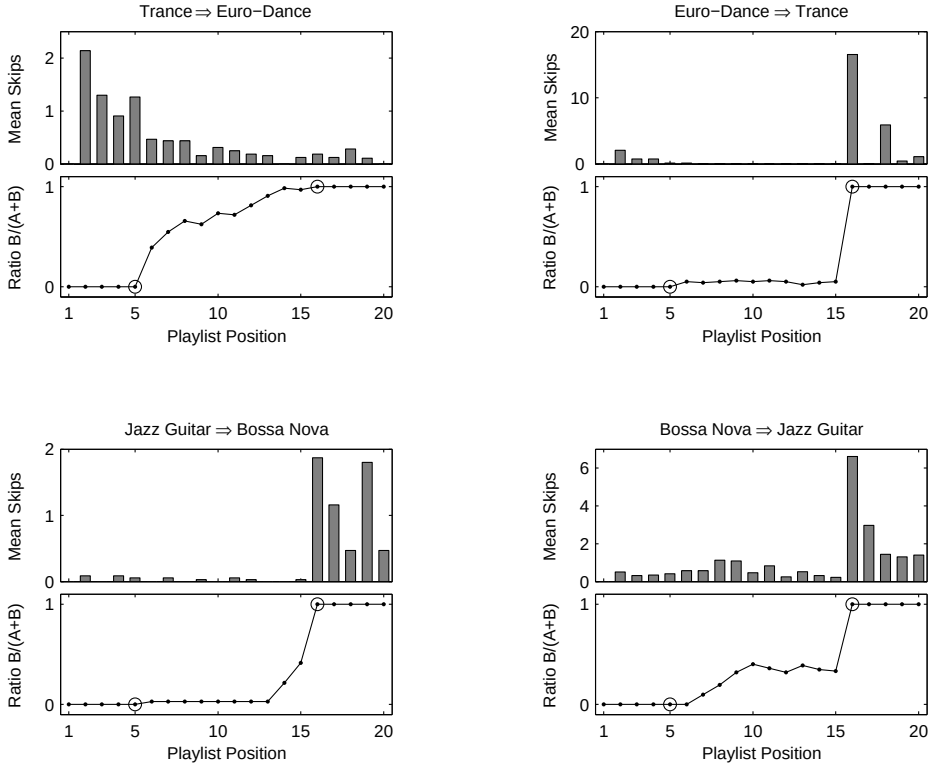


Figure 3: Average skips and genre ratio per playlist position for heuristic D in UC-3. The genre ratio is 0 if only genre A (the genre of the seed) is played and 1 if only genre B (destination genre) is played. The circle marks the last and first song which is forced to be from a specific genre.