AN ADAPTIVE DISTANCE MEASURE FOR SIMILARITY BASED PLAYLIST GENERATION

D. Gärtner, F. Kraft

Universität Karlsruhe
interACT
Am Fasanengarten 1, Karlsruhe, Germany

*T. Schaaf**

Carnegie Mellon University
interACT
5000 Forbes Ave, Pittsburgh, PA, USA

ABSTRACT

Nowadays, a large part of all music ever recorded is digitally available and due to MP3 already ten thousands of songs can be carried around on a mobile device. Intelligent automatic song selection is more and more required alternatively to random selection or manual playlist generation. We propose a system, that generates playlists including songs similar to accepted ones, discarding songs similar to rejected ones, where similar refers to timbre. Additional adaptivity is achieved with a user-adaptive distance function which in our case requires modeling features separately. After a seed-song (which is the first accepted song) is given by the user, the distance function is used by a song selection strategy to select songs. Minimal user feedback is collected with a skip button that is pressed to directly jump to the next song and explicitly reject the current one while acceptance is implicitly given by listening to a song.

***Index Terms*—** playlist generation, user adaptation, skipping behavior, music similarity, music information retrieval

1. INTRODUCTION

Music data is difficult to organize, because there is no common system everybody would agree to. This is why methods to retrieve music from this large set of data are necessary.

One aspect of a huge collection of music pieces is that not all songs may fit the taste or are currently of interest to a listener. The optimal solution would be that the audio-player, e.g. a small mobile device, would know by itself what kind of music someone wants to listen next. However, the kind of music which is appropriate at the moment can depend on so many factors like current mood or the environment. These factors are difficult or impossible to model and therefore an indirect way to observe this is to suggest music and learn form the user reaction. This allows a form of playlist generation that adapts to the listeners needs, which is the topic of this paper.

To generate a playlist, the system uses a seed-song, which is selected by the user. Thenceforward, songs are automatically chosen and played by the system. The user rates the played songs implicitly by either skipping and thus rejecting them or accepting the choice by just listening to the complete song. Those already classified songs are taken into account when tracks are selected and to update the distance function.

Playlist generation fulfilling given constraints is investigated in [1, 2, 3], but constraints have to be defined first and rich meta-data needs to be accessible. In [4] playlist generation is performed using music similarity based on meta-data. Complete playlists can be scored by the user to help improving the generation process for further playlists. Meta data and acoustic similarity are used in [5]. User tests have shown that compiling playlists using the help of similarity leads to playlists with the same quality than manually created ones but in much less time. Rich meta data is used in, e.g., the Music Genome Project[1], where experts analyze each featured song for close to 400 attributes. In Gracenote playlist[2] information about genres, era of the recording artist, release date or geographical origin (taken from their database, the former CDDB, currently storing information about over 66.000.000 songs) are used to draw relationships between songs. Last.FM[3] is building relationships based on collaborative filtering techniques involving the songs played by the millions of their users. Ordering all songs from a collection based on audio similarity is done in [6] with the help of traveling salesman algorithms. A playlist generation system working only on audio similarity is proposed in [7], where the playlist consists of the n closest neighbors to a given seed-song. In a similar approach [8] to the one proposed in this publication only acoustic similarity and user-feedback is used to compile playlist.However, a static distance function is used. [9] shows, that feature subsets lead to significant improvement in perceived music similarity.

2. METHODOLOGY

Figure 1 shows an overview of the components in our system. From each song of the database, features are extracted. The distribution of each feature is statistically modeled, and a feature-based distance is computed between each pair of songs on a certain feature. Based on those feature-based distances, a global distance between songs can be computed. This global distance is used by the song selection strategy to select the next song that will be played, involving also information which of the already played songs were accepted and which were rejected by the user. Once in a while, the global distance function is adapted to the user, using the already played songs.

We want to generate playlists, that include 19 songs similar to the seed-song. We then have a playlist l with a total of 20 accepted songs which comes up to the capacity of one CD. s_l, the number of rejected songs which is equal to the number of times, the skip button was pushed during the generation process of a playlist is to be minimized.

2.1. Database

Music similarity systems are hard to evaluate automatically, since music similarity sensation is highly subjective. For this work, genre affiliation is used to define similarity, which is a common principle to evaluate large parameter spaces and was shown to be reasonable

*Now with Toshiba Research Europe Ltd, Cambridge, United Kingdom

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

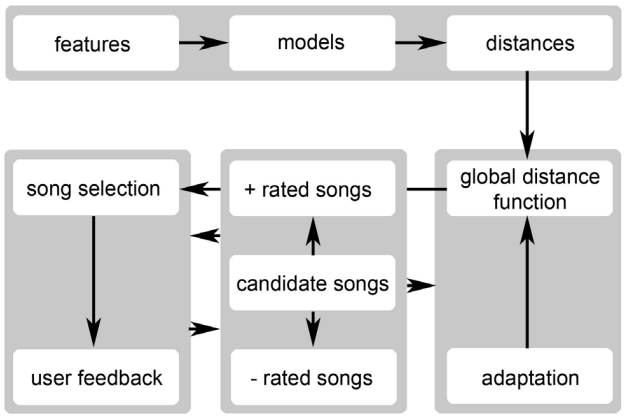

Fig. 1. System architecture overview.

genre	classical CL	electronic EL	jazz_blues JB
#songs	320	115	26
genre	metal_punk MP	rock_pop RP	world WO
#songs	45	101	122

Table 1. Distribution of the genres.

[10]. Two songs are assumed to be similar exactly if they belong to the same genre. The database consists of the songs from the 2004 ISMIR Genre Classification Contest Trainingset[4]. The distribution of the songs can be seen in Table 1.

2.2. Evaluation

In a complete evaluation run, 703 playlists are generated, using each song from the genres $\mathcal{G} = \{\mathrm{CL}, \mathrm{EL}, \mathrm{MP}, \mathrm{RP}, \mathrm{WO}\}$ (Table 1) as seed-song once. Songs from JB are not taken as seed-songs, since this genre is underrepresented for including it in our evaluation measures, but they still remain in the database and can be proposed during a playlist generation process. Two two-stage measures are defined to evaluate the performance of the system. In the first stage the arithmetic mean (median) of the skip values s_l of playlists created with seed-songs from a certain genre is determined. If the song selection is using a nondeterministic algorithm, this procedure is repeated R times and the mean (median) values are replaced by the arithmetic mean of the resulting values of those R repetitions. In the second stage the arithmetic mean over those genre based values is calculated. The arithmetic mean based measure is called average skip measure (ASM), the median based measure is called median skip measure (MSM). This two-stage process assures that the impact of a class on the measure is independent of the size of the class.

$$\mathrm{ASM} = \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}\frac{1}{R}\sum_{i=1}^{R}\frac{1}{|S_g|}\sum_{s\in S_g}\mathrm{playlistSkipN}(s)$$

$$\mathrm{MSM} = \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}\frac{1}{R}\sum_{i=1}^{R}\underset{s\in S_g}{\mathrm{median}}(\mathrm{playlistSkipN}(s))$$

playlistSkipN(s) returns s_l, the number of skips needed to create the playlist l using s as seed-song, S_g denotes all the songs from genre g.

[4] http://ismir2004.ismir.net/genre_contest/index.htm

	CL	EL	MP	RP	WO	(A/M)SM
mean	43.3	120.2	307.8	137.2	113.5	144.4
median	42.7	118.6	307.8	135.8	112.5	143.5

Table 2. Random baseline (genre means / medians, ASM, MSM).

Outliers, which can be songs that are very dissimilar to other songs of their genre, or songs that are very similar to songs of another genre, in some cases can lead to very high skip values for a certain playlist. Furthermore, outliers can also be caused by problems during modeling, If the ASM value is much higher than the MSM value, this is a sign for the influence of outliers.

ASM (MSM) values are no abstract values. ASM specifies a weighted average of how often the skip button needs to be pressed to reach a playlist with 20 accepted songs and MSM specifies a weighted average of how often the skip button needs to be pressed at most in half of the cases. Lower values are preferable.

Table 2 shows the genre means (median means) with the resulting ASM (MSM) value for a system using random song selection on our used database. Since random selection is nondeterministic we used $R = 100$. All the song selection strategies described later in this paper use deterministic algorithms so R is set to 1.

2.3. Feature Extraction and Model Estimation

FFmpeg[5] is used to convert the 128 kbit MP3 files from the dataset to 16kHz 16bit mono PCM files. All the used features are calculated on power spectral frames of 30s from the middle of the audio signal, which are derived using 16ms windows with 10ms shift. Mel Frequency Cepstral Coefficients (MFCCs) and Mel Frequency LDA Coefficients (MFLCs) are used as basic feature. MFLCs are calculated similar to MFCCs except that the cosine transform is replaced [11] by an Linear Discriminant Analysis transform (LDA). With respect to train the LDA transform, each song represents its own class, which is then more compact and more distant from the other classes in the transformed feature space. We are using 40 Mel bins and extract 21 MFCCs, from which the first one is disregarded, and 20 MFLCs (those with the largest eigenvalues). In addition to the basic feature, we are using the following LDA-sidefeatures $f_x \in F$, each of them is extracted in 16 linearly spaced nonoverlapping subbands and then LDA transformed: centroid (10), bandwidth (4), skewness (5) and kurtosis (1), all described in [12], and flatness (4), crest factor (4), Shannon entropy (6) and Renyi entropy of order 2 (13) as described in [13], where the number in brackets denotes the number of kept coefficients after LDA which is equal to the number of Eigenvalues above an empirically determined value of 1.1. A featureset including MFLCs and all the sidefeatures will be denoted FULL.

For each feature of each song, one multivariate normal distribution is estimated. We are using full covariance matrices.

2.4. Song Selection

For song selection, two algorithms introduced in [8] are used.

S1: The song with the smallest distance (see section 3) to any of the accepted songs is played.

S2: From all the candidate songs (those that have not been proposed by the system) a subset is chosen, containing exactly all songs who's nearest neighbor in the set of the already classified songs was accepted. From this subset, the one with the smallest distance to this nearest neighbor is chosen. If there is no song in the subset, the song

[5] http://ffmpeg.mplayerhq.hu/

with the best ratio of distance to the nearest accepted neighbor and distance to the nearest rejected neighbor is chosen. This expresses the desire to have a song close to accepted and far apart from rejected songs.

3. THE DISTANCE FUNCTION

The distance function is used by the song selection strategy to determine the next song.

3.1. Likelihood Ratio Hypothesis Test

A likelihood ratio hypothesis test is used to compute distances between models. The null hypothesis, that observations from both models were generated by a joined model is divided by the alternate hypothesis, that both feature vector sequences were generated by different models. As described in [14], for computation of the likelihoods of the hypotheses it is assumed that both density functions have the same mean. Since all of the models have been trained with the same amount of samples, the computation can be simplifi ed. We use the negative log-likelihood ratio as distance:

$$d(\mathcal{A},\mathcal{B}) = log|\frac{1}{2}\Sigma_{\mathcal{A}} + \frac{1}{2}\Sigma_{\mathcal{B}}| - \frac{1}{2}log|\Sigma_{\mathcal{A}}| - \frac{1}{2}log|\Sigma_{\mathcal{B}}|,$$

where $\Sigma_{\mathcal{A}}$ and $\Sigma_{\mathcal{B}}$ are the covariance matrices of song $\mathcal{A}$ and $\mathcal{B}$ respectively. In [15] the choice of the likelihood ratio hypothesis test as distance between models is explained in detail.

3.2. Calculation of song dissimilarity

Let f_x be feature x, F the complete feature set, $d_{f_x}(\mathcal{A},\mathcal{B})$ is the variance normalized value for the described likelihood ratio hypothesis test using the models of song $\mathcal{A}$ and $\mathcal{B}$, and w_x the binary weight, announcing whether f_x is currently used or not. We use the variance normalization to transform the distances based on features with different ranges to make distances on different features comparable to eachother. The global distance $D(\mathcal{A},\mathcal{B})$ between $\mathcal{A}$ and $\mathcal{B}$ is then computed by:

$$D(\mathcal{A},\mathcal{B}) = \sum_{k=1}^{|F|} w_k d_{f_k}(\mathcal{A},\mathcal{B})$$

$D(\mathcal{A},\mathcal{B})$ then is also variance normalized over all the different feature combinations, to make combinations using less features comparable to combinations using many features.

3.3. User adaptation of the global distance function

Initially, all the binary weights are set to 1. A brute force approach is used to adapt the binary weights to the musical taste of the user, using the +rated songs $\mathcal{P}$ and the -rated songs $\mathcal{N}$. For each possible feature combination, three sums are computed:

- s^+, the sum over the distances of each $p \in \mathcal{P}$ to it's nearest neighbor in $\mathcal{P}$.
- s^-, the sum over the distances of each $n \in \mathcal{N}$ to it's nearest neighbor in $\mathcal{N}$.
- $s^\pm$, the sum over the distances of each $p \in \mathcal{P}$ to it's nearest neighbor in $\mathcal{N}$.

Then, $\text{score}(\omega) = w_p s^+ + w_n s^- - w_{pn} s^\pm$ is the score of a feature combination ω, where w_p, w_n, and w_{pn} are the adaptation weights. The score of every possible feature combination is computed and the feature combination with the smallest score is chosen - we want s^+ and s^- to be small since the two classes should be compact, and $s^\pm$ to be large since the two classes should be far apart from eachother. The weights and the adaptation interval v, which denotes how often the skip button has to be pushed until the next adaptation is performed, are denoted in an adaptation vector $< w_p, w_n, w_{pn}, v >$.

	CL	EL	MP	RP	WO	ASM/MSM
S1						
MFCC	6.6	40.4	17.9	15.4	73.9	30.8/24.4
MFLC	3.0	40.3	16.0	12.6	65.5	27.5/21.6
FULL	0.9	51.8	11.6	7.0	76.7	29.6/28.2
S2						
MFCC	2.6	18.5	19.3	17.7	26.2	16.9/11.2
MFLC	1.3	19.8	23.6	13.9	23.0	16.3/11.4
FULL	0.5	17.2	5.6	5.0	32.4	12.1/10.8

Table 3. MFCCs, MFLCs and FULL set, selection S1 and S2 (genre means, ASM, MSM).

4. RESULTS

In table 3, the performance of MFCCs and MFLCs is compared to the performance of FULL. Comparing MFCCs with MFLCs, MFLCs lead to better playlists (ASM) for both selection strategies.For S2, a performance increase from 16.3 to 12.1 ASM is observed for adding sidefeatures to MFLCs. The sidefeatures obviously capture additional useful aspects of the audio signal, although they (like MFLCs) all describe the spectral shape of the signal. However, performance decreases for S1 from 27.5 to 29.6 ASM. S1, which basically leads to worse results than S2, seems to be inapplicable when using feature spaces also including useless features, since it cannot learn from negative examples. The difference between ASM and MSM is rather small for the FULL set compared to the performance of the MFCC / MFLC sets for both selection strategies, which is a sign for the FULL set being less vulnerable to outliers. It is not surprising that the WO genre performs worst for almost all confi gurations. WO, loosely described by songs being non-western music is already divers by definition. But since users that don't want to listen to songs besides songs from the world genre, but on the other hand like the complete bandwidth of that genre are hardly to imagine, this should be no drawback of the system.

Table 4 shows the results on the complete set for different adaptation weights and selection S1, Table 5 shows the results in the complete set for different adaptation weights and selection S2. It can be seen that for $< 0, 0, 1, 5 >$ the performance worsens a lot for both S1 and S2 (ASM). All the other weighting combinations lead to better results for S1 and to comparable results for S2 (ASM). The best performance is achieved by setting all three weights to 1. The so far best result of 12.1 ASM for S2 and the FULL set can be further improved to 11.9. The positive impact of the adaptation is larger for S1 compared to S2. S2 already incorporates information about $\mathcal{N}$ while S1 does not. The influence of the adaptation interval can be seen in Figure 2, using the FULL set and S2. Adapting the weights every 10 skips gives the lowest average skip rate while adapting too early leads to small training data for adaptation. The later adaptation is performed, the later useless features are excluded. For many playlists of genres with small mean skip values, adaptation is not performed, since there are less than v skips needed. Adapting after every 10 skips leads to an ASM of 11.4, which is the overall best reported result.

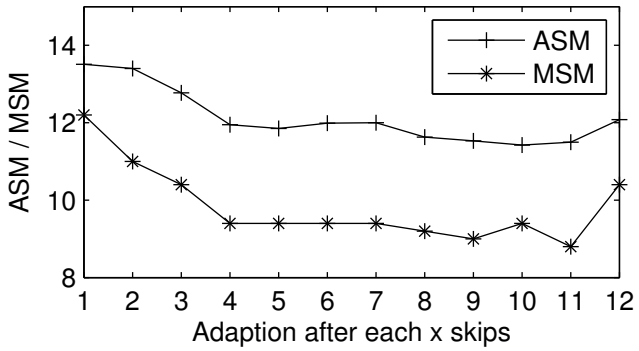

Fig. 2. Adaptation of feature combination weights in different intervals, S2, $< 1, 1, 1, x >$.

5. CONCLUSION AND OUTLOOK

We described our system for playlist generation for music.It uses a single button for user feedback which is skipping the current played music. We investigated MFCC and MFLC features to be used as basic features and found that MFLCs outperform MFCCs (ASM). Features which are usually computed over the whole spectrum were extracted from sub-bands and then transformed and reduced in dimensionality with an LDA. Using side features in addition to the basic feature MFLCs improves the performance of the system for S2, the better of the two strategies. Adaptation of the distance function can further improve the quality of the playlists. Using random selection on the used database leads to an ASM of 144.4 which is reduced to 11.4 for the best configuration. The task and the way the evaluation is performed is vulnerable to outliers. We assume that adding features which describe other musical properties than the spectral shape will not only improve the overall quality of the playlists but also allow to increase the gain from adaptation. Further gain is assumed to be observed when using a variable adaptation interval. Starting with a large v to gain enough training data for adaptation, and then perform adaptation more often after more skipped songs become available. In future we would further like to test the system with users to investigate the usability.

6. ACKNOWLEDGMENTS

We want to thank Alex Waibel and interACT for supporting this work.

	CL	EL	MP	RP	WO	ASM/MSM
noadap	0.9	51.8	**11.6**	**7.0**	76.7	29.6/28.2
<0,0,1,5>	0.8	87.5	31.4	14.0	85.1	43.8/34.0
<0,1,0,5>	0.8	**36.8**	15.5	8.9	50.7	22.5/**20.0**
<0,1,1,5>	**0.6**	39.4	16.2	7.6	**47.3**	22.2/20.2
<1,0,0,5>	0.7	38.6	14.3	8.0	53.2	23.0/20.4
<1,0,1,5>	**0.6**	48.2	20.4	**7.0**	57.2	26.7/21.6
<1,1,0,5>	0.7	37.3	15.0	7.8	50.6	22.3/20.2
<1,1,1,5>	**0.6**	38.6	14.5	7.5	48.1	**21.8**/20.2

Table 4. Different adaptation weights, S1 (genre means, ASM, MSM).

weights	CL	EL	MP	RP	WO	ASM/MSM
noadap	**0.5**	**17.2**	**5.6**	**5.0**	32.4	12.1/10.8
<0,0,1,5>	0.6	64.7	23.7	9.6	56.0	30.9 / 20.8
<0,1,0,5>	**0.5**	17.5	6.6	7.6	27.1	**11.9** / 9.6
<0,1,1,5>	**0.5**	20.3	10.2	7.1	29.0	13.4 / 10.2
<1,0,0,5>	**0.5**	19.3	6.6	6.1	28.9	12.3 / 10.8
<1,0,1,5>	**0.5**	20.3	11.7	7.0	29.7	13.8 / 9.8
<1,1,0,5>	**0.5**	17.8	7.6	6.9	28.5	12.2 / 10.2
<1,1,1,5>	**0.5**	19.0	8.0	6.9	**24.9**	**11.9 / 9.4**

Table 5. Different adaptation weights, S2 (genre means, ASM, MSM).

7. REFERENCES

[1] M. Alghoniemy and A. Tewfik, "Personalized music distribution," in *Proc. ICASSP*, 2000, vol. 6, pp. 2433–2436.

[2] M. Alghoniemy and A. Tewfik, "A network flow model for playlist generation," in *Proc. ICME*, 2001.

[3] J.-J. Aucouturier, "Scaling up music playlist generation," in *Proc. ICME*, 2002.

[4] S. Pauws and B. Eggen, "Pats: Realization and user evaluation of an automatic playlist generator," in *Proc. ISMIR*, 2002.

[5] S. Pauws and S. van der Wijdeven, "User evaluation of a new interactive playlist generation concept," in *Proc. ISMIR*, 2005, pp. 638–643.

[6] T. Pohle, E. Pampalk, and G. Widmer, "Generating similarity-based playlists using traveling salesman algorithms," in *Proc. DAFx*, 2005.

[7] B. Logan, "Content-based playlist generation: Exploratory experiments," in *Proc. ISMIR*, 2002.

[8] E. Pampalk, T. Pohle, and G. Widmer, "Dynamic playlist generation based on skipping behavior," in *Proc. ISMIR*, 2005, pp. 634–637.

[9] A. S. Lampropoulos, D. N. Sotiropoulos, and G. A. Tsihrintzis, "Individualization of music similarity perception via feature subset selection," in *Proc. SMC*, 2004, pp. 552–556.

[10] E. Pampalk, *Computational Models of Music Similarity and their Application in Music Information Retrieval*, Ph.D. thesis, Technische Universität Wien, Fakultät für Informatik, 2006.

[11] T. Eisele, R. Haeb-Umbach, and D. Langmann, "A comparative study of linear feature transformation techniques for automatic speech recognition," in *Proc. ICSLP*, 1996, vol. 1, pp. 252–255.

[12] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," Tech. Rep., IRCAM, Paris, France, 2004.

[13] A. Ramalingam and S. Krishnan, "Gaussian mixture modeling using short time fourier transform features for audio fingerprinting," in *Proc. ICME*, 2005.

[14] H. Gish, M.-H. Siu, and R. Rohlicek, "Segregation of speakers for speech recognition and speaker identification," in *Proc. ICASSP*, 1991.

[15] Daniel Gärtner, "User adaptive music similarity with an application to playlist generation," M.S. thesis, Universität Karlsruhe, ITI Waibel, http://isl.ira.uka.de/fileadmin/publication-files/gaertner_DA.pdf, 2006.