Combining Audio-based Similarity with Web-based Data to Accelerate Automatic Music Playlist Generation

Peter Knees[1], Tim Pohle[1], Markus Schedl[1], and Gerhard Widmer[1,2]
[1]Department of Computational Perception, Johannes Kepler University Linz, Austria
[2]Austrian Research Institute for Artificial Intelligence (OFAI)
peter.knees@jku.at, tim.pohle@jku.at, markus.schedl@jku.at, gerhard.widmer@jku.at

ABSTRACT

We present a technique for combining audio signal-based music similarity with web-based musical artist similarity to accelerate the task of automatic playlist generation. We demonstrate the applicability of our proposed method by extending a recently published interface for music players that benefits from intelligent structuring of audio collections. While the original approach involves the calculation of similarities between every pair of songs in a collection, we incorporate web-based data to reduce the number of necessary similarity calculations. More precisely, we exploit artist similarity determined automatically by means of web retrieval to avoid similarity calculation between tracks of dissimilar and/or unrelated artists. We evaluate our acceleration technique on two audio collections with different characteristics. It turns out that the proposed combination of audio- and text-based similarity not only reduces the number of necessary calculations considerably but also yields better results, in terms of musical quality, than the initial approach based on audio data only. Additionally, we conducted a small user study that further confirms the quality of the resulting playlists.

Categories and Subject Descriptors

H.5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing

General Terms

Algorithms

Keywords

Automatic Playlist Generation, Web-based Artist Similarity, Music Similarity, Music Information Retrieval, Traveling Salesman Problem

MIR'06, October 26–27, 2006, Santa Barbara, California, USA.

1. INTRODUCTION

Structuring and organizing large music repositories is a non-trivial problem. Nevertheless, the constant upward trend of electronic music distribution requires sophisticated methods for accomplishing these tasks automatically. The rapidly growing field of Music Information Retrieval is a direct response to these and related demands.

As in traditional Information Retrieval, one of the central challenges is to find well suited content descriptors. Several approaches exist to extract features for describing music. The most common is to analyze the audio signal directly (for an overview, see e.g. [15]). A complementary approach is to analyze texts that concern on music or musical artists and their work. For convenience, this meta-data is usually taken from the web by invoking a search engine, e.g. querying with the artist's name. The obtained web pages can be used to calculate, for example, artist similarity by applying text-retrieval methods [22, 5, 8, 6].

This paper contributes to the current state of the art by presenting a new and effective way to combine these distinct sources of information for efficient music playlist generation. Music playlists are predefined sequences of music pieces. In many cases, it is comfortable for the user to have playlists that fit the current situation, e.g. parties, jogging, or a romantic dinner. While all previously presented approaches only make use of audio-based features, we incorporate additional web-based features to reduce the number of audio-based similarity calculations as well as improving the quality of the generated playlists. We will demonstrate the applicability by extending an interface for mobile music players, which we call the "wheel" [18].

This paper is organized as follows. In the next section, we give a brief overview of existing approaches to combine audio-based features with web-based features and existing (audio-based) approaches to automatic playlist generation, including our own "wheel"-interface. In Section 3, we review the initial methodology from [18] and present our extension. In Section 4, we compare our extension to the original technique in terms of complexity and performance. In the last section, we draw conclusions and point out future directions.

2. RELATED WORK

First, we will review three (to the best of our knowledge, all existing) approaches that combine audio-based and web-based data. Second, we give an overview of automatic playlist generation systems, among them our own "wheel", which will be used to demonstrate the effectiveness of our combination technique throughout the remainder of this paper.

2.1 Audio-based and Web-based Combination

In [22], audio-based as well as web-based genre classification are used for the task of style detection on a set of 5 genres with 5 artists each. Combining the predictions made by both methods linearly yields perfect overall prediction for all test cases. In [4], audio-based track similarity is linearly combined with web-based artist similarity to obtain a new similarity measure. In [9], we augment an interface to music collections with terms obtained from the web. The interface consists of a three-dimensional island landscape that places the musical pieces according to their sound similarity. The user can freely navigate in this virtual environment. The exploration is supported by presenting terms on the landscape that are related to the audio content in that region and the corresponding artists. Thus, it provides semantic feedback based on music.

2.2 Automatic Playlist Generation

The problem of playlist generation is treated as a network flow problem in [1]. Given a start track and an end track in a song collection, the algorithm finds a path (of user-defined length) through the network satisfying user-defined constraints. Each piece is labeled with a number of boolean attributes representing arbitrary aspects of the music. [2] presents a more efficient approach for handling various types of meta-data. According to user-defined constraints, the meta-data of each track is transformed into a cost function. The playlist is constructed by iteratively optimizing an initial randomly chosen playlist with regard to the cost function. In [11], labeled tracks are not assumed, since the playlist generation algorithm is based on a music similarity function (cf. [12]), which can be computed automatically. Several approaches are evaluated for producing a playlist of given length for a given start track. In [7], a highly interactive user interface that facilitates music exploration and playlist generation is presented. The user can grab pieces of music from similarity-based flows of tracks to create playlists. The interface presented in [21] relies on an underlying map (regardless whether automatically or manually created) and puts a focus on social interaction at playlist creation.

For creating playlists on mobile devices, [14] incorporates Self-Organizing Maps (SOM) [10]. SOMs are used to cluster similar pieces of music and are visualized by means of Smoothed Data Histograms, cf. [17]. This visualization leads to a 2-dimensional representation of the collection which is inspired by geographical maps. Playlists can be defined intuitively by drawing paths on the map.

From our point of view, users are seldom capable of paying much attention to the management of their playlists while *en route*. Therefore, we have presented an approach that makes a complete music collection easily accessible through a simple wheel [18]. This approach automatically organizes a music collection into a large circular playlist by applying a Traveling Salesman Algorithm on the calculated music similarity. If such an algorithm succeeds in finding the best tour, the generated playlist satisfies the constraint that consecutive tracks are maximally similar on average. The whole playlist and thus the whole collection is accessible with only one circular controller – the "wheel" (cf. Figure 1).

In this paper, we present an extension to the "wheel". While the original approach involves the calculation of acoustical similarities between every pair of songs in a collection, we incorporate web-based data to reduce the number of necessary similarity calculations. To this end, we retrieve information about artists from the web, which we use to assess the similarity of artists. It turns out that by using this combination, we can reduce the number of necessary similarity calculations considerably and also improve the basic approach that is based on audio only.

Figure 1: A screenshot of our Java applet "Traveller's Sound Player".

3. METHODOLOGY

With our playlist generation approach, we aim at maximizing the average similarity between consecutive tracks in a playlist, and thus, obtaining playlists containing large sections of consistent music. A resulting playlist can be interpreted as a projection of the whole collection onto one dimension. The collection is arranged around a circular wheel. Coherent areas of different musical styles should emerge naturally around the wheel; these can then be directly accessed via a simple wheel turn.

In this section, we explain our approach to generate one large playlist consisting of all tracks from the collection by modeling a Traveling Salesman Problem (TSP). First, we describe the functionality of the original approach that operates on a full matrix of musical distances obtained from an audio-based similarity function. This comprises the calculating audio similarity and applying a TSP algorithm. Subsequently, we describe our enhancement based on incorporating web-derived features. Since the web provides knowledge and opinions of a large number of people, also "cultural" aspects are covered by this approach.

3.1 Audio-based Similarity

In this work, we decided to use the well-established algorithmic outline proposed by Aucouturier and Pachet in [3], since it outperforms most other audio-based approaches, cf. [15]. This approach is based on Mel Frequency Cepstral Coefficients (MFCCs) computed on short-time audio segments. As proposed in [3], we calculated 19 MFCCs. Each track is then represented as a Gaussian Mixture Model (GMM) of the distribution of MFCCs. The similarity of two tracks is calculated by sampling from one GMM and determining the probability that these samples would have been generated by the other track's GMM. Similarity calculation for each pair of tracks in a collection of size n results in an $n \times n$ matrix, on which we model a Traveling Salesman Problem.

Note that our approach is not restricted to this particular similarity measure. It also smoothly integrates with other audio-based measures, such as [12, 16, 13].

3.2 Traveling Salesman Problem

Finding an optimal solution for the Traveling Salesman Problem is NP-hard, which implies that there is no known algorithm that calculates the exact result fast for large data sets. Many heuristics have been proposed that approximate the correct result. In the initial approach, we selected four of them for evaluation. Here, we will limit the evaluation to the *Minimum Spanning Tree* algorithm since it performed well in previous experiments. Furthermore, it is capable of dealing with edges of infinite costs. This is particularly important for our proposed extension (see below). For convenience, we call this algorithm *MinSpan* in this paper. This algorithm (e.g. [20]) is evaluated although it makes the assumption that the triangle inequality is fulfilled, which is not the case on the data we use[1]. First, a minimum spanning tree is found with a standard algorithm (Kruskal algorithm) in $O(n \cdot \log n)$, with n being the number of edges. Afterwards, a depth-first search is performed on the minimum spanning tree, and a tour is constructed by connecting the nodes in the order they are first visited during the depth-first search. Thus, at least in the current implementation, some transitions, i.e. edges not contained in the minimum spanning tree, are introduced into the route irrespective of their actual costs.

3.3 Web-based Similarity

Since perception of music is also influenced by information other than the pure audio signal, e.g. cultural, social, historical, and/or contextual aspects, this kind of information should not be neglected when generating playlists. On the one hand, it may help to avoid stylistic confusions sometimes made by audio-based similarity measures. On the other hand, a priori information can be utilized to reduce the effort to be expended for playlist creation. An easily accessible source for this kind of information is the World Wide Web since it incorporates many people's knowledge and opinions. Thus, we use the names of the artists contained in the collection to complement the audio-based similarity measure with cultural knowledge by applying web-information retrieval techniques. The used technique has been successfully applied to the task of genre classification and yielded promising results for direct artist similarity [8]: For each artist, we search the web with Google. The query string consists of the artist's name as an exact phrase extended by the keyword *music*. We retrieve 50 of the top-ranked web pages for each query, remove all HTML tags, and use common English stop word lists to remove frequent terms. For computational efficiency, we also remove all terms that do not occur on at least c pages over all artists. We choose the threshold c such that about 10,000 terms remain.

[1]Many TSP algorithms require the distance measure d between the nodes to satisfy the triangle inequality $d(ac) \leq d(ab) + d(bc)$ for all triples a, b and c. On the audio similarity matrix constructed from our test data, the triangle inequality does not hold in about five percent of the cases when comparing randomly chosen direct and alternative edges. On data that satisfies the triangle inequality, *MinSpan* produces a tour that is guaranteed not to be longer than twice the optimum tour.

Folk–Rock(4) Rap(4) Jazz(1) Punk–Rock(1)	Electronica(2)	Electronica(5)	Electronica(1)	Electronica(16) Acid Jazz(1)
Folk–Rock(1) Italian(1)	Electronica(1)	Acid Jazz(1)		Acid Jazz(1) Electronica(1)
Italian(3) Electronica(1)		Reggae(2) Italian(1)		Rap(2) A Cappella(1) Acid Jazz(1) Electronica(1)
Punk–Rock(4) Electronica(1)	Rap(4)		Blues(1)	Jazz(3)
Electronica(12) Punk–Rock(1)	Rap(1) Electronica(1)	Celtic(2) Reggae(1)	Celtic(3) A Cappella(1)	Jazz(5) Bossa Nova(4) Blues(3) A Cappella(2) Rap(1)

Figure 2: An example 5×5 SOM trained on the web-based features of our second test collection. For reasons of lucidity, the corresponding genres instead of the individual artists are listed. The values in parenthesis represent the number of artists per genre mapped to the respective unit.

For each artist a and each term t appearing in the retrieved pages, we count the number of occurrences tf_{ta} (term frequency) of term t in documents related to a. Furthermore, we count df_t the number of pages the term occurred in (document frequency). These are combined using the term frequency $\times$ inverse document frequency ($tf \times idf$) function [19]. The term weight per artist is computed as

$$w_{ta} = \begin{cases} (1 + \log_2 tf_{ta}) \log_2 \frac{N}{df_t} & \text{if } tf_{ta} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where N is the total number of pages retrieved.

As a result, each artist is described by a vector of term weights. The weights are normalized such that the length of the vector equals 1 (Cosine normalization). This removes the influence of the retrieved web pages' length. Using this representation, similarities between artists can be derived, for example, by calculating Euclidean distances. However, we prefer another way of combining the two similarity sources, which is elaborated next.

3.4 Combining Both Approaches

Currently, only two approaches that directly combine audio-based track similarity and web-based artist similarity exist (see Section 2.1). In both, classification results or similarities are simply linearly combined into a single result. In our case, a linear combination is not useful since it does not reduce the number of necessary computations. Furthermore, this kind of combination only modifies the audio-based distances by applying the artist distances directly to their corresponding tracks, creating another proximity matrix that is biased toward tracks from the same artist. Since we aim at exploiting web-based artist similarity to reduce the effort to be made for audio similarity calculation, we use the additional artist information to prohibit similarity calculations of songs that are unlikely to be similar. To this end, the similarity of artists is taken as an indicator for the similarity of their songs. Consequently, we only calculate distances of tracks by similar artists. All the other pairwise similarities are assumed to be infinite, i.e. for the TSP, there is no connection between the corresponding "cities". Thus, these transitions are not available when building the minimum spanning tree.

This method not only improves the "quality" of the generated playlists, as we shall see below, also the number of necessary calculations decreases considerably. To avoid introducing an "artist filter", i.e. grouping all tracks from one artist together in the playlist, we redefine artist similarity using a SOM [10] that is trained on the set of web-based term weight vectors. We define two artists to be similar if they are mapped to the same unit or to adjacent units of the SOM, i.e. their Manhattan distance is less than or equal to 1. In cases where the SOM contains units without neighbors, or more generally, where the neighborhood graph of the SOM units is disconnected, we introduce additional transitions to assure the producibility of a minimum spanning tree for the *MinSpan* algorithm. To this end, we find those units from disconnected parts whose model vectors have minimum distance in feature space and add transitions between all artists contained on these units. We iterate this step until all parts of the map are connected. A SOM resulting from our second evaluation collection is depicted in Figure 2.

We utilize the SOM as a convenient method to determine the binary similarity measure. Other approaches comprise finding the *k-Nearest Neighbors* or defining a threshold to find a variable number of similar artists. We decided to train a SOM as this does not require to define such parameters. However, in our case, the chosen size of the SOM has an important impact on the number of audio calculations to be carried out. We will systematically evaluate the impact of the SOM size on the resulting playlist in the next section. The goal is to find a setting which disposes of as many transitions as possible between songs unlikely to be similar while leaving enough options for the TSP algorithm to find a route.

4. EVALUATION

To evaluate our approach, we pursue two strategies. First, we perform a quantitative evaluation by carrying out systematic experiments and measuring long-term consistency of the playlists. Second, we present the results of a small user study which was carried out to evaluate the practical usefulness of the approach. Finally, we tackle the theoretical background concerning runtime complexity.

4.1 Test Collections

For our experiments, we used two in-house collections. The first contains 3 456 tracks (by 339 artists) assigned to 7 general, quite evenly distributed genres: Classical (14.7%), Dance (15.0%), Hip-Hop (14.5%), Jazz (13.6%), Metal (14.9%), Pop (11.6%), and Punk (15.6%). The minimum number of tracks per artist is 1, the maximum 317.

In the second collection, also very specific genres are present to demonstrate the applicability of the approach on non-standard music. The second collection contains 2 545 tracks (by 103 artists) assigned to 13 genres: A Cappella (4.4%), Acid Jazz (2.7%), Blues (2.5%), Bossa Nova (2.8%), Celtic (5.2%), Electronica (21.1%), Folk Rock (9.4%), Italian (5.6%), Jazz (5.3%), Metal (16.1%), Punk Rock (10.2%), Rap (12.9%), and Reggae (1.8%). The minimum number of tracks per artist is 8, the maximum 61.

4.2 Quantitative Evaluation

We evaluate the improvement of our extension compared to the original, audio-only approach. Our focus is on the reduction of necessary similarity calculations. In addition, we require a measure to quantify the "quality" of a playlist. We decided to consider the long-term consistency according to genre, i.e. how clearly defined the type of music is in a certain region of the wheel. We assume that a user would expect coherent parts of music in an angle of maximally 45 degrees (for a detailed discussion see [18]). This corresponds to one eight of the whole playlist (432 tracks for the first, 318 tracks for the second collection).

4.2.1 Long-Term Consistency

To estimate consistency, the *Shannon entropy* of the genre distribution is calculated on playlist sequences comprising one eighth of the playlist: It is counted how many of n consecutive tracks belong to each genre. The result is normalized and interpreted as a probability distribution, on which the Shannon entropy is calculated. The Shannon entropy is defined as

$$H(x) = -\sum_{x} p(x) \log_2 p(x) \tag{2}$$

with $\log_2 p(x) = 0$ if $p(x) = 0$. This value is averaged over the whole playlist (i.e. each track of the playlist is chosen once as the starting track for a sequence of length 432 or 318, depending on the used collection). For lack of a more precise qualitative evaluation criterion, we assume that lower entropy values indicate a more intuitive arrangement of the music.

4.2.2 Impact of SOM Size

To study the impact of the SOM size on the number of calculations and the long-term consistency, we performed a series of experiments with quadratic SOMs in the range of 1×1 to 10×10. Every experiment has been carried out five times. In the following, we only consider the average value for each setting, which is sufficient to illustrate the general trend. Figure 3 illustrates the fractions of necessary audio similarity calculations when employing our acceleration technique on the number of calculations necessary without using our approach. Furthermore, the corresponding long-term entropy values for the different SOM sizes are displayed.

An obvious finding is that for the amount of transitions, the first collection has consistently higher values than the second. In contrast, for the long-term entropy, the values for the second collection are always higher. This is no surprise, since the first collection contains three times as many artists than the second but only half as many genres. The leftmost points (SOM size 1×1) reflect the initial approach without any combination with web data. Since all artists are contained on a single unit, distances between all tracks

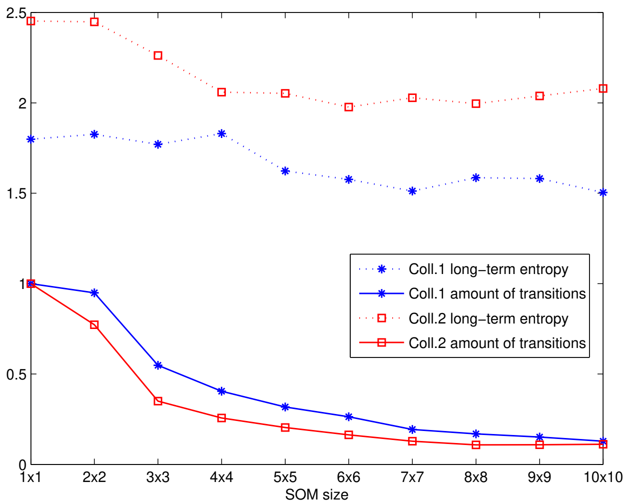

Figure 3: Number of audio-based calculations necessary (fraction of the maximum) and long-term entropy values for different SOM sizes (average of 5 runs).

must be calculated. For increasing SOM sizes, the number of similarity calculations decreases considerably[2]. For the second collection, using a small SOM with 4 units (2×2) reduces the effort to 77%, a 3×3 SOM drastically reduces the amount of required calculations to 35%. For a 6×6 SOM about 16%, for even larger SOMs only 10% of calculations remain. Interestingly, for the 10×10 and 9×9 SOMs the value is slightly higher than for the 8×8 SOM. We assume this to be caused by the fact that the number of units equals roughly the number of data items (artists). Thus, no useful similarity can be produced by the Manhattan distance technique and similarity is mainly defined by our connection approach (see Section 3.4). For the first collection, a similar behavior can be observed (without the small increase at the end), considering the fact that due to the higher number of artists, larger SOMs have a higher impact.

Taking a look at the development of the long-term entropy in Figure 3 reveals that (for one exception) incorporating web data leads to lower entropy values and thus more consistent playlists with respect to genre. To get a better impression of the impact of the SOM size on playlist quality, Figure 4 visualizes the genre distribution of a typical playlist for each SOM size.

From the obtained results, we can conclude that for 103 artists, a SOM with 36 units is best suited, leading to approximately 3 artists per unit in average. As far as we can see for the other collection (339 artists), again, a SOM (10×10) whose number of units equals about one third of the number of artists leads to the best results. Choosing such a SOM size, for each artist, similarities for all tracks by 15 similar artists (including self-references) have to be calculated in average.

[2]Compared to the number of similarity calculations, we can neglect the time required to compute the SOMs.

4.3 Subjective Evaluation

To prove the practical usefulness of the approach, we carried out a small user study with 10 test persons. Using the first test collection, we created a playlist from which we extracted 10 short sequences with length 10 each. To this end, we randomly chose a starting point on the "wheel" and consecutively extracted the 10-track-playlists at intervals of 36 degrees. Each test person had to rate each playlist with respect to overall musical consistency on a scale ranging from 1 ("totally inconsistent") to 5 ("completely smooth transitions"). Results can be found in Table 1.

playlist no.	1	2	3	4	5	6	7	8	9	10
avg. rating	4.8	3.6	2.6	2.7	3.8	4.1	3.6	5.0	5.0	2.3

Table 1: Average rating values for the 10 test playlists. The average rating over all playlists equals 3.92.

It can be seen that 7 out of 10 playlists have been given over 3 points on average. Furthermore, the raw data reveals that, out of the total 100 ratings, 33 were top-ranked (5 points) and 29 gained 4 points. This shows that the majority of playlists automatically generated by our approach is considered useful and very consistent by users.

4.4 Runtime Complexity

Finally, we want to give theoretical support to the empirical evidence presented in the preceding section. To this end, we make some assumptions about the average music collection. Let a be the number of artists and n be the overall number of tracks in the collection. Our assumptions are:

- A collection contains more than two artists ($a > 2$).
- The number of tracks per artist is bounded by a constant t_{max}.
- The distribution of musical styles in a collection is well-balanced. Thus, on the SOM, each unit is assigned the same number of artists. By choosing an appropriate SOM size depending on a, we can keep the number of artists per unit, denoted by u, constant. For each artist, this leads to $s = 5 \cdot u$ similar artists (including self-reference).

Based on these assumptions we can estimate the number of song similarities that have to be calculated as $O(a \cdot s \cdot t_{max}^2)$. Since s and t_{max} can be estimated with upper bounds, the number of calculations depends on a instead of n^2. More precisely, instead of runtime $O(n^2 \cdot \log\ n^2)$ for the audio only approach, we can reduce runtime complexity to $O(a \cdot \log\ a)$ for the *MinSpan* algorithm.

However, since the size of the SOM is a function of a, we have runtime of $O(a^2)$ in the training phase of the SOM. In the training step, for a given number of iterations, the distances between all units' model vectors ($f(a)$) and all data items (a) are calculated. Nevertheless, with our extension we can reduce complexity from $O(n^2 \cdot \log\ n^2)$ to $O(a^2)$. In practice, the time required to compute the SOM is negligible compared to the calculation time for the track similarities.

In terms of the behavior of our approach, we can get further interesting insights when considering two extreme scenarios. The first one regards a collection that contains only music by one artist. Incorporating web-based information has no effect at all. The second scenario is a collection of tracks where each track is performed by a different artist, e.g. a collection of "one-hit wonders". In this case, the artist similarity is directly applicable as track similarity. However, it is questionable whether this is desirable, especially since it also requires quite an effort to acquire the necessary information for all artists.

This leads to the final consideration regarding runtime – the time necessary for the web retrieval. Even though retrieval of the web pages is linear in the number of artists, downloading and processing 50 related pages is time consuming (approximately 30 seconds per artist). Nevertheless, the proposed approach can be modified such that other meta-information are incorporated and exploited to calculate similarities, e.g. manually assigned artist information.

5. CONCLUSIONS

We have presented an approach to accelerating automatic playlist generation. This is accomplished by incorporating musical artist similarity based on web data. Besides the achieved improvements in runtime complexity, we could also show that the combination of audio-based similarity measures with web-based data improves the resulting playlists in terms of stylistic consistency. We further proved the usefulness of the created playlists with a small user study.

An important aspect of this paper is the bridging of the gap between audio-based track similarity and web-based artist similarity. Indeed, we have shown that these complementary approaches are very powerful in conjunction. Thus, we can conclude that the proposed combination technique can also be used to reduce complexity and improve the results of similar tasks like automatic music recommendation. For future work, we aim at reducing the effort to be made to obtain web-based features.

6. ACKNOWLEDGMENTS

This research is supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project number L112-N04 and by the Vienna Science and Technology Fund (WWTF) under project number CI010 (Interfaces to Music). The Austrian Research Institute for Artificial Intelligence acknowledges financial support by the Austrian ministries BMBWK and BMVIT.

7. REFERENCES

[1] M. Alghoniemy and A. Tewfik. A Network Flow Model for Playlist Generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'01)*, Tokyo, Japan, August 22-25 2001.

[2] J-J. Aucouturier and F. Pachet. Scaling Up Music Playlist Generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'02)*, Lausanne, Switzerland, August 26-29 2002.

[3] 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.

[4] S. Baumann. *Artificial Listening Systems: Modellierung und Approximation der subjektiven Perzeption von Musikähnlichkeit.* PhD thesis, Technical University of Kaiserslautern, 2005.

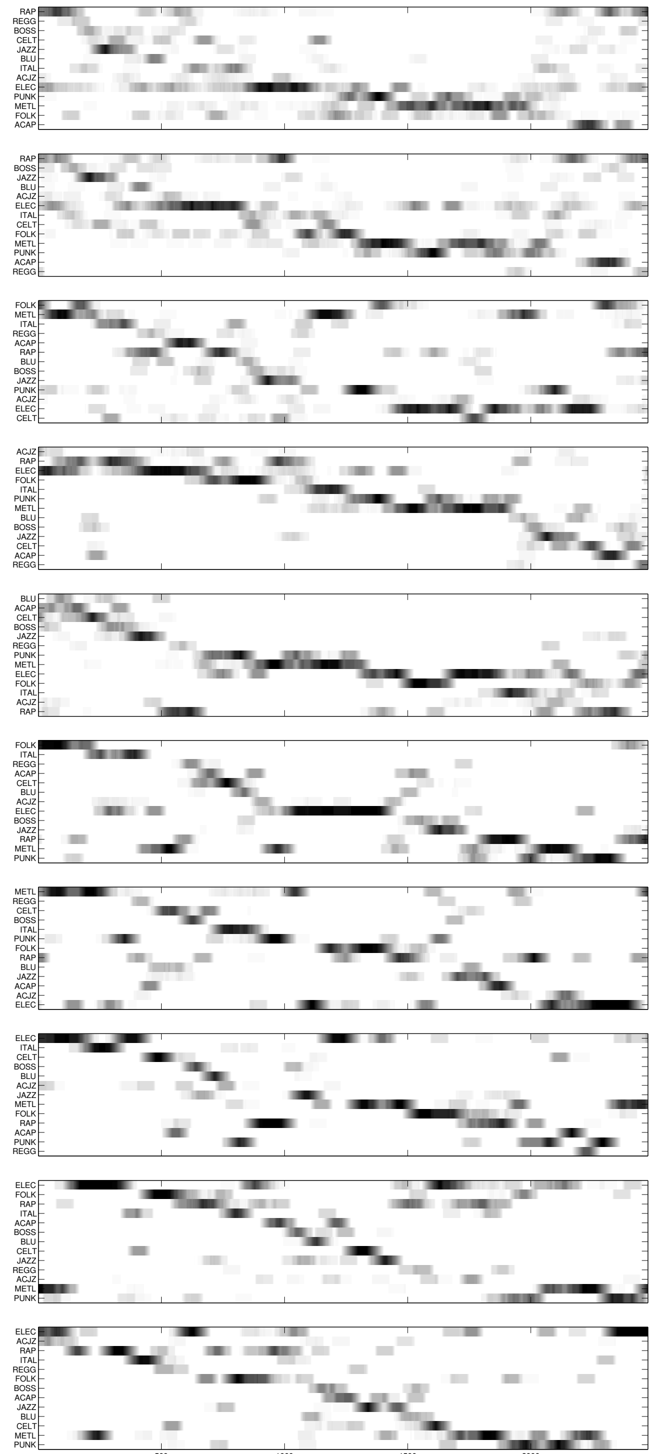

Figure 4: Genre distributions in playlists generated on our second collection, descending from 1×1 (audio only) to 10×10 SOM. Dark segments indicate a high agglomeration of tracks from the corresponding genre. The genres are ordered by the index where most pieces of that genre are accumulated. Thus, in the optimum case a playlist would tend to result in a diagonally descending sequence of black bars.

[5] S. Baumann and O. Hummel. Using Cultural Metadata for Artist Recommendation. In *Proceedings of the 3rd WedelMusic Conference*, September 2003.

[6] W.W. Cohen and W. Fan. Web-Collaborative Filtering: Recommending Music by Crawling The Web. *WWW9 / Computer Networks*, 33(1-6):685–698, 2000.

[7] M. Goto and T. Goto. Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, September 2005.

[8] P. Knees, E. Pampalk, and G. Widmer. Artist Classification with Web-based Data. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, pages 517–524, Barcelona, Spain, October 2004.

[9] P. Knees, M. Schedl, T. Pohle, and G. Widmer. An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In *Proceedings of the ACM Multimedia 2006*, Santa Barbara, California, USA, October 2006.

[10] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[11] B. Logan. Content-Based Playlist Generation: Exploratory Experiments. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, Paris, France, October 2002.

[12] B. Logan and A. Salomon. A Music Similarity Function Based on Signal Analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'01)*, Tokyo, Japan, August 22-25 2001.

[13] M. Mandel and D. Ellis. Song-Level Features and Support Vector Machines for Music Classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[14] R. Neumayer, M. Dittenbach, and A. Rauber. PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, 2005.

[15] E. Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, March 2006.

[16] E. Pampalk, A. Rauber, and D. Merkl. Content-Based Organization and Visualization of Music Archives. In *Proceedings of the ACM Multimedia*, pages 570–579, Juan les Pins, France, December 1-6 2002.

[17] E. Pampalk, A. Rauber, and D. Merkl. Using Smoothed Data Histograms for Cluster Visualization in Self-Organizing Maps. In *Proceedings of the International Conference on Artifical Neural Networks (ICANN'02)*, pages 871–876, Madrid, Spain, August 2002. Springer.

[18] T. Pohle, E. Pampalk, and G. Widmer. Generating Similarity-based Playlists Using Traveling Salesman Algorithms. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*, pages 220–225, Madrid, Spain, September 20-22 2005.

[19] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[20] S.S. Skiena. *The Algorithm Design Manual*. Springer-Verlag, New York, Department of Computer Science, State University of New York, Stony Brook, NY 11794-4400, 1997.

[21] I. Stavness, J. Gluck, L. Vilhan, and S. Fels. The MUSICtable: A Map-Based Ubiquitous System for Social Interaction with a Digital Music Collection. In *Proceedings of the 4th International Conference on Entertainment Computing (ICEC'05)*, Sanda, Japan, 2005.

[22] B. Whitman and P. Smaragdis. Combining Musical and Cultural Features for Intelligent Style Detection. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, Paris, France, October 2002.