Large-Scale Music Exploration in Hierarchically Organized Landscapes Using Prototypicality Information

[deepTune]

Markus Schedl, Christian Höglinger, Peter Knees

Department of Computational Perception
Johannes Kepler University
Linz, Austria
markus.schedl@jku.at

ABSTRACT

We present a novel user interface that offers a fun way to explore music collections in virtual landscapes in a game-like manner. Extending previous work, special attention is paid to scalability and user interaction. In this vein, the ever growing size of today's music collections is addressed in two ways that allow for visualizing and browsing nearly arbitrarily sized music repositories. First, the proposed user interface `deepTune` employs a hierarchical version of the Self-Organizing Map (SOM) to cluster similar pieces of music using multiple, hierarchically aligned layers. Second, to facilitate orientation in the landscape by presenting well-known anchor points to the user, a combination of Web-based and audio signal-based information extraction techniques to determine cluster prototypes (songs) is proposed. Selecting representative and well-known prototypes – the former is ensured by using signal-based features, the latter by using Web-based data – is crucial for browsing large music collections. We further report on results of an evaluation carried out to assess the quality of the proposed cluster prototype ranking.

Categories and Subject Descriptors: H.5.2 [User Interfaces]: Auditory, Graphical user interfaces H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities

General Terms: Algorithms

Keywords: music information extraction, user interface, human–computer interaction, unsupervised learning

1. BACKGROUND AND RELATED WORK

Steadily growing sizes of digital music collections, both in the private and the commercial area, necessitate intelligent user interfaces [18] to make the vast amount of music available to all. Methods for exploring music repositories by means beyond simple text-based interfaces[1] are thus gaining more and more popularity. According to [33], music retrieval systems to access music collections can be broadly categorized with respect to the employed *query formulation* method into *direct querying*, *query by example*, and *browsing* systems. The approach presented in the paper at hand uses the modality of browsing to retrieve music from a possibly huge repository.

One group of algorithms that aims at offering the user a means of intelligently exploring music collections is *music recommender systems*, e.g., [4]. Such a system is usually built by first deriving features from various sources, such as tags obtained via game playing [9, 32], artist term profiles extracted from Web pages [2, 12], or RSS feeds [5]. Subsequently, a similarity measure between artists or between songs is applied to the feature vectors. The resulting similarity estimates are then used to recommend music similar to a given input song or artist. As an alternative or in addition, collaborative filtering techniques [3] may be used for or incorporated into the recommendation.

1.1 Intelligent User Interfaces to Music

Another category of approaches to transcend traditional, text-based ways of music retrieval is *intelligent user interfaces* (IUI) [18] to explore music collections. The `deepTune` application presented here is one example of such an IUI. Others include [10], where songs are represented as discs that drop down from various taps (corresponding to different moods) and can be arranged and combined to form playlists. [21] and [22] present user interfaces to explore music collections according to different similarity dimensions (acoustic similarity as well as similarity derived from term profiles of artist-related Web pages). [25] proposes an interface that organizes a music collection in a large circular playlist by approximating the solution to a Traveling Salesman Problem that is defined by the collection's audio similarity matrix. Similar pieces of music are therefore found in similar regions along a disc visualization, which can be accessed via a wheel. Several extensions of this interface have been proposed, mainly to improve the user's orientation within the music collection: for example, [27] presents an implementa-

[1]The majority of today's music playback devices still employs the traditional step-wise search scheme for `artist - album - track`.

tion on a mobile device that incorporates Web-based tags to facilitate navigation, and [8] presents an approach that automatically structures the playlist hierarchically.

Most closely related to the `deepTune` interface is the `nepTune` application [13] that builds upon the "Islands of Music" (IoM) metaphor [19, 20]. According to this metaphor, similar music pieces are visualized via islands with homogeneous acoustic properties (for example, a "Classical" island, a "Heavy Metal" island, and so on). Depending on the (musical) distance between these islands, they are separated by large oceans or small sand banks. IoM uses the unsupervised learning algorithm *Self-Organizing Map* (SOM) [14] to determine music clusters and subsequently approximates the distribution of the collection's data items over the map by a *Smoothed Data Histogram* (SDH) [24]. In [13] a three-dimensional extension of the IoM is presented. The clusters are determined according to acoustic similarity. In addition, terms and images extracted from Web pages are presented to describe the regions of the map. While navigating through the landscape, the songs closest to the user's current position are played simultaneously. [17] presents a similar three-dimensional user interface. The authors, in contrast, use a metaphor different from the "Islands of Music". Their height map algorithm produces inverse heights, compared to the IoM approach, i.e., agglomerations of music pieces are located in valleys, and the clusters are not separated by oceans, but by hills. This technique resembles the *U-matrix* visualization [34] of the Self-Organizing Map. Moreover, user adaptation is supported by allowing the user to build or destroy separating hills. In this case, the similarity measure is adapted accordingly.

A drawback of the interface proposed in [13] is that it does not scale beyond some hundreds of songs, because of computational limitations and restricted visualization space. The `deepTune` application, in contrast, extends [13] in that it clusters the given music collection in a hierarchical manner, thus allows to visualize arbitrarily sized music collections. Given today's large amounts of tracks in personal music repositories, scalable, intelligent interfaces are of particular importance. We hence used a hierarchical clustering algorithm. The resulting multi-layer visualization requires various extensions to guide the user in her exploratory music discovery. These visual extensions, as well as the algorithm to find representative cluster prototypes that we had to elaborate, are detailed in the following section.

2. TECHNICAL FOUNDATIONS

The `deepTune` system makes use of various techniques from the fields of music information research, Web mining, and unsupervised learning. First, acoustic features (*Fluctuation Patterns*) are extracted from the audio signal of the input songs. Based on these features, a clustering algorithm then organizes the collection. Since `deepTune` should be able to visualize arbitrarily sized music collections, we opted for a hierarchical version of the Self-Organizing Map (SOM) clustering approach. Well-known prototypes for each cluster are subsequently determined by a Web-based popularity detection technique, the popularity ratings of which are combined with acoustic features in order to find music pieces that are representative for the cluster, but also popular enough to be of help for the majority of music listeners.

2.1 Signal-based Audio Features

Acoustic features are computed according to the approach proposed in [23]. The *Fluctuation Patterns* (FP) describe rhythmical properties as they represent a music piece's distribution of re-occurring beats over different frequency bands and modulation frequencies (at different bpm).

To compute the FP features, first each track is sliced into 6-second-pieces. Then, the audio signal of selected pieces is transformed into the frequency domain by applying a *Fast Fourier Transform* (FFT) [6]. Subsequently, the frequency/magnitude scale is transformed into the *psychoacoustic Bark scale* [35], according to which frequency values are binned into perceptually equidistant "critical bands". Since the human ear is not equally responsive to all frequencies, a *perceptual model of the human auditory system* [31] is applied to account for the disproportionately high impact of low frequencies and the disproportionately low impact of high frequencies. Furthermore, *spectral masking* effects, i.e., the occlusion of quieter sounds if two or more sounds of similar frequency co-occur, are taken into account [28]. The modified Bark scale values are then transformed into the perceptually linear *Sone scale* [30]. The final *Fluctuation Pattern* of a piece of music is then obtained by computing the *Discrete Cosine Transform* (DCT) [1] of the modified power spectrum of each 6-second-slice, subsequent emphasizing perceptually important periodicities, and aggregating the resulting rhythm periodicity representations for the entire track by computing the median over all slices.

Figure 1 illustrates the Fluctuation Patterns of highly different music pieces. The FP depicted on the upper left belongs to a techno track with dominant bass beats around 120 and 240 bpm (corresponding to 2 and 4 Hertz, respectively). The piece on the lower left is a quiet song dominated by calm voices. The one on the upper right is an rock song with a poignant female voice. The piece on the lower right is a piano sonata with a clearly horizontal characteristic, without any activations in the lowest frequency bands.

Applying the FP computation results in a 1,200-dimensional feature vector representation (20 critical bands times 60 periodicity bins) for each piece of music. To decorrelate redundant feature dimensions and improve performance the feature vectors of all songs are compressed to 120 dimensions using *Principal Components Analysis* (PCA) [11]. This representation is then input into the clustering algorithm.

2.2 Clustering

To group similarly sounding music pieces (according to the Fluctuation Patterns), we reimplemented and extended the *Growing Hierarchical Self-Organizing Map* (GHSOM) clustering algorithm presented in [7]. The standard SOM is a neural network model that performs a non-linear mapping from a high-dimensional data space into a low-dimensional (usually two-dimensional) visualization space while preserving topological properties, i.e., data items that are similar in the feature space are mapped to similar positions of the visualization space. A SOM is described as a set of map units U arranged in a rectangle. Each map unit u_i is assigned a model vector m_i with same dimensionality as the data space. During training the model vectors are gradually adapted to better represent the input data X. The map unit's model vector closest to a data item x is referred to as x's "best-matching unit" and is used to represent x on the map.

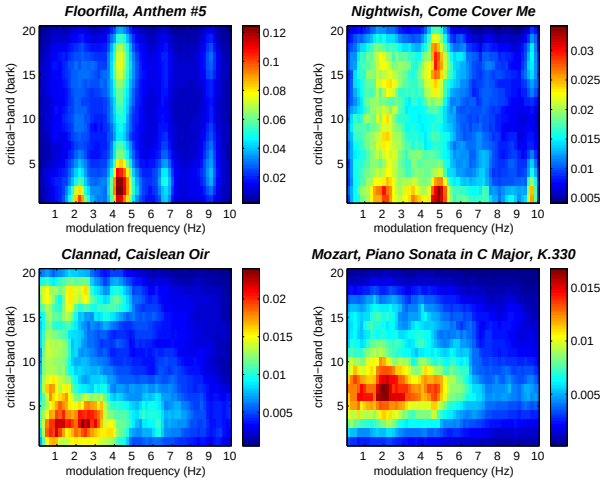


Figure 1: Fluctual Pattern visualization for different pieces of music.

In practice the standard SOM approach is limited in the number of data items that can be visualized. We therefore opted for the GHSOM approach that automatically adapts the structure of the SOM during training. Starting with a standard SOM of size 2×2, in each iteration step during training, the *mean quantization error* of each map unit mqe_i and of the whole SOM $mmqe$ is calculated according to Formulas 1 and 2, respectively. V_i represents the Voronoi set of map unit u_i, i.e., the set of all data items for which u_i is the best-matching unit, m_i is the model vector that describes u_i, and $|X|$ is the cardinality of the input data set.

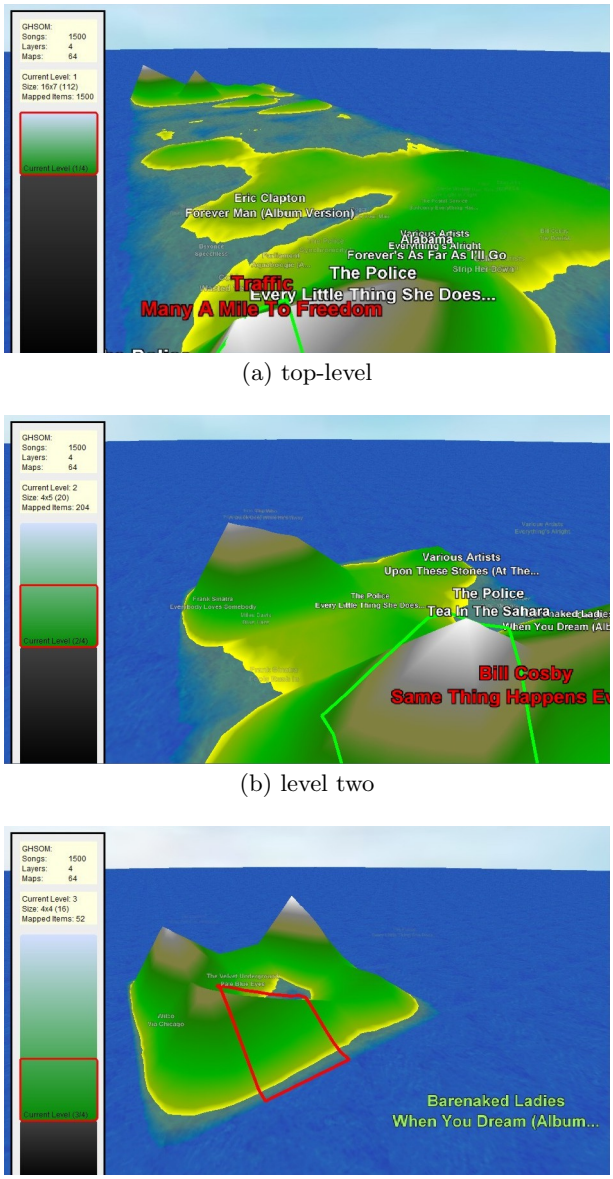
$$mqe_i = \frac{1}{|V_i|} \cdot \sum_{j \in V_i} \|x_j - m_i\| \tag{1}$$

$$mmqe = \sum_i \frac{|V_i|}{|X|} \cdot mqe_i \tag{2}$$

The parameter τ_m controls the size of individual SOMs, whereas τ_u regulates the depth of the GHSOM. In our experiments we empirically set $\tau_m = 0.5$ and $\tau_u = 0.25$.
To enforce a quadratic layout of the (sub-)SOMs, we further introduced a restricting parameter for the ratio between the number of rows and columns (set to 0.5). Moreover, we modified the algorithm in that SOMs representing less than 10 data items are not further expanded. This circumvents creating a lot of very sparse sub-level SOMs.

2.3 Cluster Prototype Selection

Depicting the labels of all music pieces mapped to the highest-level-SOM would yield tremendous visual overload when real-world collections consisting of tens of thousands of tracks are processed. An easy solution to this problem is to determine representative prototypes for each map unit u_i by selecting a number of data instances closest to m_i. Although this is a mathematically sound solution, the resulting prototypes are often not very popular, therefore unknown to most users, and thus of limited help for their orientation. As an alternative, we propose the following prototype selection algorithm that builds upon [8]. Prototypical music pieces for a map unit u_i are determined by combining Web-based popularity estimation and the pieces' audio-based distance to the respective map unit's model vector m_i.



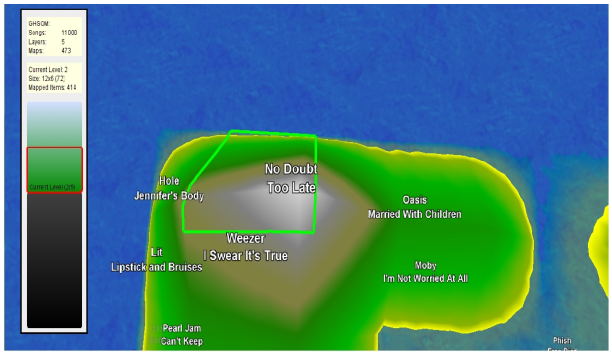
(a) top-level

(b) level two

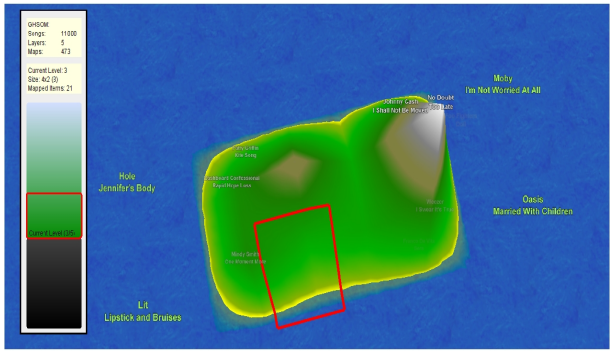
(c) level three

Figure 2: `deepTune`'s visualization on different levels of the GHSOM-tree.

First, we estimate the popularity of each artist in the collection by obtaining page counts from `Google` for queries of the form `"artist name" music review`. Based on the page-count-values, we define an artist ranking according to Formula 3, where $pc(a)$ is `Google`'s estimate for artist a's number of Web pages and $norm(\cdot)$ scales the values to the range $[1, 5]$. The audio signal-based part of the ranking function is given in Equation 4, where x is the feature vector



(a) upper level



(b) lower level

Figure 3: `deepTune` interface with anchor points.

corresponding to the music piece under consideration, $\|\cdot\|$ is the Euclidean distance, and $norm(\cdot)$ is a normalization function that shifts the range to $[1, 2]$. Finally, the artist-based popularity ranking and the track-based ranking of audio similarity to the model vector under consideration m_i are combined (Formula 5), and the pieces with highest $r(x)$ value are selected as prototypes for u_i.

$$r_w(a) = norm_{[1,5]}(log_{10}(pc(a))) \tag{3}$$

$$r_s(x) = norm_{[1,2]}\left(\frac{1}{1 + \ln(1 + \|x - m_i\|)}\right) \tag{4}$$

$$r(x) = r_s(x) \cdot r_w(a) \tag{5}$$

An evaluation of this ranking technique has been conducted as well. The results are presented in Section 4.

3. USER INTERFACE

To get a first impression of the `deepTune` application, Figures 2(a), 2(b), and 2(c) depict the visualization resulting from a sample collection, respectively, on the top, on the second, and on the third level in the GHSOM-tree. The height of the landscape is derived from the voting matrix of the SDH [24], i.e., it roughly corresponds to the number of pieces mapped to each map unit. These height values are further encoded as colors, according to a color map used for topographical maps. The resulting landscape can be regarded as "Islands of Music". Each map unit is assigned a number of most representative tracks, the labels of which are depicted above the corresponding unit. Note that `deepTune` employs linear initialization of the SOM and batch training; the resulting maps are hence stable for a constant data set.

User interaction within the `deepTune` environment is provided either by using a mouse or a game pad. The interface supports panning, rotating, and adjusting the viewpoint angle. Furthermore, a "quick zoom" function facilitates swift orientation in large landscapes.

The currently played track is highlighted via flashing of its label. When moving through the landscape, a green rectangle around a map unit illustrates that the map unit can be expanded, i.e., lower-level SOMs do exist. A red rectangle denotes map units that cannot be expanded further. Upon pressing a button, the user "dives" into the surrounded map unit to the lower-level SOM. To prevent the user from getting lost in deep SOM hierarchies, sub-SOMs are placed into their higher-level context by showing the prototypes of their parents' neighboring map units, which serve as anchor points for better orientation. Figure 3 illustrates this concept by highlighting different anchor points (the green labels on the lower screenshot). Note that the layout of the anchor points within the lower level resembles the layout of the prototypes of the surrounding map units in the upper level (upper image). Moreover, a navigation bar illustrates the current depth in the GHSOM-tree and reveals further information on the visualization (e.g., the total number of SOMs and the size of the currently displayed SOM in terms of map units and represented data items). Furthermore, an "escape" function immediately brings the user back to the top-level SOM.

In order to alleviate the visual clutter that would arise from depicting the whole Voronoi set of each map unit, the number of prototypical pieces shown per map unit is limited. The actual number of prototypes shown for map unit u_i is determined by Equation 6, where V_i is the Voronoi set of map unit u_i, and m is the maximum number of prototypes per unit to be displayed.

$$np(u_i) = \left\lceil \frac{ln(|V_i|)}{ln(max_j(|V_j|))} \cdot m \right\rceil \tag{6}$$

3.1 Implementation Aspects

The audio features are calculated and compressed via PCA using the `CoMIRVA` framework [26] for music information retrieval and visualization. Also the artist popularity estimation builds upon Web retrieval functionality provided by `CoMIRVA`. We extended the framework by our variant of the GHSOM implementation.

The `deepTune` application itself is implemented in `Java`, using the libraries `Xith3D` for graphics processing and `OpenAL` as audio API. `deepTune` has been tested on a real-world music collection of about 48,000 songs, which is a subset of a digital music retailer's catalog.

4. EVALUATION

The selection of suitable cluster prototypes is essential for the usability of `deepTune`. To assess the quality of the proposed ranking approach (cf. Section 2.3), we compared the results obtained by our ranking function with play count data extracted from the music information system `last.fm` [15] for the same artists/tracks. To retrieve the play count

data we used last.fm's API [16]. Note that we refrained from directly using last.fm's play counts in deepTune since building a Web crawler and simple page counts estimator is feasible without relying on commercial, proprietary systems.

Two evaluation steps have been performed. First, the pure Web-based artist ranking function $r_w(a)$ has been evaluated in order to assess its significance for the complete ranking function. Second, $r(x)$, the combined signal- and Web-based ranking function for particular sets of songs located on a specific map unit has been evaluated.
To illustrate the evaluation results, we calculated *Spearman's rank correlation coefficient* [29] and used a scatter plot.

4.1 Evaluation of $r_w(a)$

A list of 7,723 unique artist names has been extracted from our test database of 47,757 songs. For each artist name, two Web requests were issued. First, the Google page count corresponding to the query "artist name" music review was obtained. Second, the artist's overall last.fm play count was retrieved. Subsequently, tie-adjusted rankings were calculated. Tie adjustment was especially necessary for the calculation of Spearman's rank correlation coefficient, as uncorrected ties distort the result. Considering the page counts and play counts of the data set used, most ties were caused by either Google returning the value 0 for page counts or last.fm returning the value -1, indicating that the artist queried is known to the system. Tied ranks were dealt with by assigning each tied item the mean of its surrounding items' ranks.

Calculating Spearman's rank correlation coefficient for the tie-adjusted rankings results in the value 0.819, which indicates a strong correlation between last.fm's play counts and Google's page counts. This correlation is further revealed in the scatter plot depicted in Figure 4. Each point represents a specific artist, the axes correspond to the respective rankings. The x-axis represents the Google-based ranking, whereas the y-axis represents the ranking based on the last.fm play counts. The higher the value, the more popular an artist is considered by the respective data source. Thus, the highly unpopular artists should be located in the bottom left corner of the plot and the highly popular artists in the top right corner. If both rankings were perfectly similar, a straight line from the point of origin to the point (7,723; 7,723) would be visible. Even though this is obviously not the case, quite a strong correlation can be spotted, as most points are aligned around such an ideal line.
The upper left portion of the plot contains nearly no data points, contrary to the lower right portion. This indicates, that there are only few artists that have a low Google page count but a rather high last.fm playcount. Thus, if an artist is known to Google, he or she is very likely to be known to last.fm as well, but not necessarily vice versa. This finding can partly be traced back to misspellings as Google is more robust in that regard.
The straight horizontal and straight vertical sequences in the lower left portion of the plot are caused by the tie-corrected values. Both sequences indicate those artists that have either a page count value of zero or are unknown to last.fm. Such sequences affect Spearman's rank correlation coefficient positively. Nonetheless, when omitting both sequences

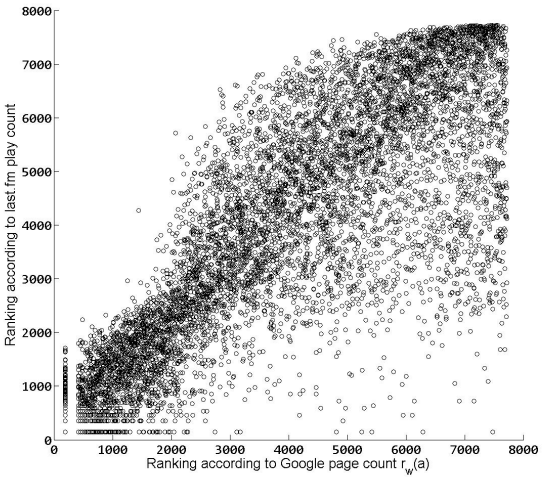


Figure 4: Scatter plot of artist ranking evaluation. Spearman's coefficient: 0.819**.**

in the calculation, a coefficient of 0.786 is attained, which is still very convincing.

In summary, a strong correlation between Google page counts, which serve as basis of deepTune's prototypicality rating, and last.fm play counts could be determined.

4.2 Evaluation of $r(x)$

The setup for the evaluation of the track-based ranking function was quite similar to the evaluation of the artist ranking. When focusing on single tracks, however, the complete ranking function, i.e., the combination of signal- and Web-based ratings, has to be evaluated. Thus, instead of the cumulated artist play counts (over all tracks), we retrieved from last.fm the play counts of specific songs, i.e., combinations of "artist name" - "track name".
First, the tie-corrected ranking function was applied to the ratings of each map unit's Voronoi set, and then the respective Spearman's rank correlation coefficient was calculated. Based on the entire collection of 47,757 songs, an overall measure as well as the results for two exemplary map units are discussed in the following.

To get a general impression of the quality of our prototype selection technique, the overall Spearman's rank correlation coefficient, averaged over all map units of the GHSOM's topmost map, was calculated according to Formula 7, where M is the number of map units, s_m is Spearman's rank correlation coefficient for map unit m having i_m data items mapped to, M denotes the total number of map units, and N the total number of data items.

$$s_{avg} = \frac{\sum_{m=1}^{M} s_m \cdot i_m}{N} \tag{7}$$

For the collection of 47,757 songs, our evaluation setting yielded an s_{avg} of 0.491, which states that the track-based ranking produced by deepTune correlates with the ranking of the corresponding last.fm play count values. Although

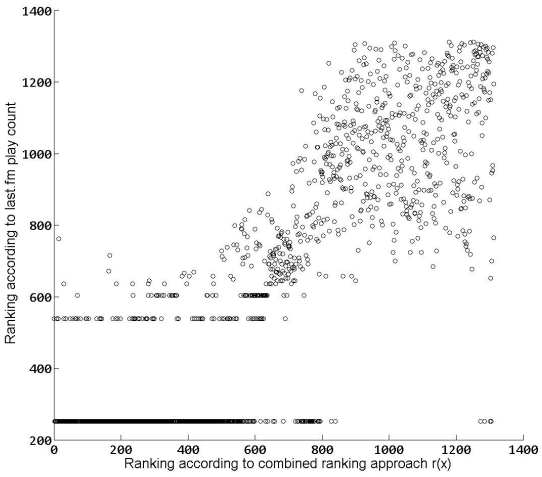


Figure 5: Track-based ranking evaluation of 1,312 **songs. Spearman's coefficient:** 0.840**.**

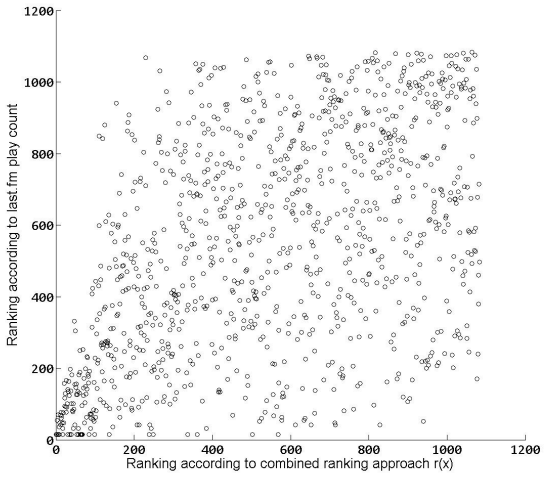


Figure 6: Track-based ranking evaluation of 1,083 **songs. Spearman's coefficient:** 0.497**.**

this result is convincing, it is not as strong as the result of the pure artist-based evaluation. That is mostly due to the fact that the track rating also incorporates a signal-based component which does not necessarily correlate with the Web-based component.

Example 1 (Strong Correlation)

This example shows a comparison of the rankings of 1,312 songs drawn from a well-populated map unit. The correlation between both rankings becomes apparent from Figure 5. Spearman's rank correlation coefficient is 0.840, which indicates a strong positive correlation.

However, the long horizontal sequence of points in the lower left corner indicates a problem with `last.fm` not recognizing the queried artist/track, which may be traced back to misspellings. In addition, the collection contains some rather unknown songs. As such a sequence considerably affects Spearman's rank correlation coefficient, its calculation was repeated without those misspelled or unknown songs. Omitting those songs results in a smaller collection of 812 tracks. For this subset, Spearman's rank correlation coefficient is 0.751, which is still remarkable.

Example 2 (Fair Correlation)

This example shows the popularity rating of another map unit with a Voronoi set of cardinality 1,083. This time the results are not quite as clear as those of Example 1. Visually, the points of the respective scatter plot, depicted in Figure 6, appear much more widespread than in the previous example. Nonetheless, when examining the plot closely, some correlation can be identified, as more densely populated areas are located in the lower left and upper right corners whereas more scarcely populated areas are present in the upper left and lower right corners.

Spearman's rank correlation coefficient for this set is 0.497, a value that indicates some correlation, but is not as distinct and convincing as in the previous example.

Interestingly, the scatter plot reveals no prominent horizontal sequence of points at the bottom of the y-axis, which means that the Voronoi set of that particular map unit contains hardly any totally unknown or misspelled songs.

5. CONCLUSIONS AND FUTURE WORK

We presented a user interface to explore large music collections in virtual landscapes. Using a hierarchical clustering approach, we partition a music collection according to rhythmical features. We further proposed a method to determine meaningful cluster prototypes, and we implemented some techniques that facilitate the user's orientation within the hierarchical visualization framework.

Future work will include integrating automated playlist generation functionality. Moreover, we would like to extend the application by "social functions". For example, in a network version of `deepTune` each user could see the music currently listened to by her friends. Users may also be able to set visual markers or indicate favorite tracks or recommendations to other users. We are further assessing methods to port `deepTune` to mobile devices. Due to system and computing limitations of current mobile platforms, calculating the audio features will likely have to be carried out on a PC.

6. ACKNOWLEDGMENTS

This research is supported by the *Austrian Science Fund* (FWF): P22856-N23, L511-N15, and Z159.

7. REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transfom. *IEEE Transactions on Computers*, 23:90–93, January 1974.

[2] S. Baumann and O. Hummel. Using Cultural Metadata for Artist Recommendation. In *Proc. of WEDELMUSIC*, Leeds, UK, 2003.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. of UAI*, San Francisco, USA, 1998. Morgan Kaufmann.

[4] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[5] O. Celma, M. Ramírez, and P. Herrera. Foafing the Music: A Music Recommendation System Based on RSS Feeds and User Preferences. In *Proc. of ISMIR*, London, UK, 2005.

[6] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[7] M. Dittenbach, D. Merkl, and A. Rauber. The Growing Hierarchical Self-Organizing Map. In *Proc. of IJCNN*, Como, Italy, 2000.

[8] M. Dopler, M. Schedl, T. Pohle, and P. Knees. Accessing Music Collections via Representative Cluster Prototypes in a Hierarchical Organization Scheme. In *Proc. of ISMIR*, Philadelphia, USA, 2008.

[9] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic Generation of Social Tags for Music Recommendation. In *Proc. of NIPS*, 2008.

[10] M. Goto and T. Goto. Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces. In *Proc. of ISMIR*, London, UK, 2005.

[11] I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, USA, 1986.

[12] P. Knees, E. Pampalk, and G. Widmer. Artist Classification with Web-based Data. In *Proc. of ISMIR*, Barcelona, Spain, 2004.

[13] 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 *Proc. of ACM Multimedia*, Santa Barbara, USA, 2006.

[14] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Germany, 3rd edition, 2001.

[15] http://last.fm (access: January 2010).

[16] http://last.fm/api (access: January 2010).

[17] D. Lübbers and M. Jarke. Adaptive Multimodal Exploration of Music Collections. In *Proc. of ISMIR*, Kobe, Japan, 2009.

[18] C. Mourlas and P. Germanakos, editors. *Intelligent User Interfaces*. Information Science Reference, Hershey, New York, USA, 2009.

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

[20] E. Pampalk, S. Dixon, and G. Widmer. Exploring Music Collections by Browsing Different Views. *Computer Music Journal*, 28(3), 2004.

[21] E. Pampalk and M. Goto. MusicRainbow: A New User Interface to Discover Artists Using Audio-based Similarity and Web-based Labeling. In *Proc. of ISMIR*, Victoria, Canada, 2006.

[22] E. Pampalk and M. Goto. MusicSun: A New Approach to Artist Recommendation. In *Proc. of ISMIR*, Vienna, Austria, 2007.

[23] E. Pampalk, A. Rauber, and D. Merkl. Content-based Organization and Visualization of Music Archives. In *Proc. of ACM Multimedia*, Juan les Pins, France, 2002.

[24] E. Pampalk, A. Rauber, and D. Merkl. Using Smoothed Data Histograms for Cluster Visualization in Self-Organizing Maps. In *Proc. of ICANN*, Madrid, Spain, 2002.

[25] T. Pohle, P. Knees, M. Schedl, E. Pampalk, and G. Widmer. "Reinventing the Wheel": A Novel Approach to Music Player Interfaces. *IEEE Transactions on Multimedia*, 9, 2007.

[26] M. Schedl, P. Knees, K. Seyerlehner, and T. Pohle. The CoMIRVA Toolkit for Visualizing Music-Related Data. In *Proc. of EuroVis*, Norrköping, Sweden, 2007.

[27] D. Schnitzer, T. Pohle, P. Knees, and G. Widmer. One-Touch Access to Music on Mobile Devices. In *Proc. of MUM*, Oulu, Finland, 2007.

[28] M. R. Schröder, B. S. Atal, and J. L. Hall. Optimizing Digital Speech Coders by Exploiting Masking Properties of the Human Ear. *Journal of the Acoustical Society of America*, 66:1647–1652, 1979.

[29] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, Boca Raton, London, New York, Washington, DC, 3rd edition, 2004.

[30] S. S. Stevens. A Scale for the Measurement of the Psychological Magnitude: Loudness. *Psychological Review*, 43(5):405–416, 1936.

[31] E. Terhardt. Calculating Virtual Pitch. *Hearing Research*, 1:155–182, 1979.

[32] D. Turnbull, R. Liu, L. Barrington, and G. Lanckriet. A Game-based Approach for Collecting Semantic Annotations of Music. In *Proc. of ISMIR*, Vienna, Austria, 2007.

[33] R. C. Veltkamp. Multimedia Retrieval Algorithmics. In *Proc. of the SOFSEM)*, Harrachov, Czech Republic, 2007.

[34] J. Vesanto. SOM-Based Data Visualization Methods. *Intelligent Data Analysis*, 3(2):111–126, 1999.

[35] E. Zwicker and H. Fastl. *Psychoacoustics, Facts and Models*, volume 22 of *Springer Series of Information Sciences*. Springer, Berlin, Germany, 2nd updated edition, 1999.