FM4 SOUNDPARK
AUDIO-BASED MUSIC RECOMMENDATION IN EVERYDAY USE

Martin Gasser
Austrian Research Institute
for Artificial Intelligence (OFAI)
martin.gasser@ofai.at

Arthur Flexer
Austrian Research Institute
for Artificial Intelligence (OFAI)
arthur.flexer@ofai.at

ABSTRACT

We present an application of content-based music recommendation techniques within an online community platform targeted at an audience interested mainly in independent and alternative music. The web platform's goals will be described, the problems of content management approaches based on daily publishing of new music tracks will be discussed, and we will give an overview of the user interfaces that have been developed to simplify access to the music collection. Finally, the adoption of content-based music recommendation tools and new user interfaces to improve user acceptance and recommendation quality will be justified by detailed user access analyses.

1 INTRODUCTION

The FM4 Soundpark is a web platform run by the Austrian public radio station FM4, a subsidiary of the Austrian Broadcasting Corperation (ORF). The FM4 Soundpark was launched in 2001 and gained significant public perception since then.

Registered artists can upload and present their music free of any charge. After a short editorial review period, new tracks are published on the frontpage of the website. Older tracks remain accessible in the order of their publication date. Visitors of the website can listen to and download all the music at no cost. Nowadays, the FM4 Soundpark attracts a large and lively community interested in up and coming music, and the radio station FM4 also picks out selected artists and plays them on terrestrial radio. At the time of writing this paper, there are 9577 tracks by 4689 artists enlisted in the online catalogue.

Whereas chronological publishing is suitable to promote new releases, older releases tend to disappear from the users' attention. In the case of the FM4 Soundpark, this has the effect of users mostly listening to music that is advertised on the frontpage, and therefore missing the full musical bandwith. To improve the accessibility of music in the FM4 Soundpark database, we proposed (1) a music recommendation system and (2) new user interfaces/visualizations that simplify accessing unknown music. Because the music available in the FM4 Soundpark was only very sparsely equipped with structured metadata (artist records can – but are not required to – be tagged with two genre labels out of a set of six), and because the active FM4 Soundpark community was considered of being yet too small for an approach based on collaborative filtering [4], we decided to implement a recommendation system utilizing a content-based music similarity measure.

2 RELATED WORK

While many research prototypes of recommendation systems/visualizations of music collections that use content-based audio similarity have been described in the literature (e.g., [5, 11, 7, 10, 6], to name just a few), very little has been reported about successful adoption of such approaches to real-life scenarios. *Mufin* [1] is advertised as a music discovery engine that uses purely content-based methods. MusicIP [2] offers the *Mixer* application that uses a combination of content-based methods and metadata to generate playlists. Bang&Olufsen recently released the Beosound 5 [3] home entertainment center, which integrates content-based audio similarity with a simple "More Of The Same Music"-user interface, that allows users to create playlists by choosing an arbitrary seed song.

3 SYSTEM OVERVIEW

As we had to integrate our system with an existing infrastructure, we placed emphasis on a decoupled design and gradual integrability of our software into the system of our industrial partner. The FM4 Soundpark recommender was implemented as a web service that offers the following functionalities: (1) synchronize the recommender with the main

[1] http://www.mufin.com/
[2] http://www.musicip.com/
[3] http://www.beosound5.com/

SMC 2009, July 23-25, Porto, Portugal

database (import/delete tracks), (2) return similar songs to a given seed song, (3) return metadata (artist/track name, artist description if available), (4) return specialized data structures for the visualizations. Most client-server communication was done with proprietary XML- or text-based protocols. The high-level web service interface was implemented in Python (using the CherryPy [4] framework), while the low-level functionality (feature extraction, database management, similarity calculation) was implemented natively in C++.

The system frontend consists of two main user interfaces: (1) an Adobe Flash MP3 player with integrated recommendation of similar songs based on pure acoustic similarity, and (2) a downloadable 3D visualization application that uses a combined similarity measure (acoustic similarity + user-defined genre labels).

3.1 Main components

As already mentioned, our software is only loosely coupled with our partner's infrastructure. Therefore, we had to define a communication protocol that can be used to trigger the necessary synchronization operations, and we had to design algorithms that are able to work on a large and constantly changing database. More precisely, the following requirements had to be met: (1) the calculation of acoustic similarities between songs must be fast and memory efficient (see 3.3) and (2) for the map generation procedure (see 3.4) a tradeoff had to be found between quality and performance.

3.2 Feature extraction

From the 22050Hz mono audio signals two minutes from the center of each song are used for further analysis. We divide the raw audio data into overlapping frames of short duration and transform them to Mel Frequency Cepstrum Coefficients (MFCC), resulting in a spectrally smoothed and perceptually meaningful representation of the audio signal. MFCCs are now a standard technique for computation of spectral similarity in music analysis (see e.g. [8]). The frame size for computation of MFCCs for our experiments was $46.4ms$ (1024 samples). We used the first 20 MFCCs for all our experiments.

3.3 Computing spectral similarity of songs

We use the following approach to music similarity based on spectral similarity. For a given music collection of songs, it consists of the following steps:

1. for each song, compute MFCCs for short overlapping frames
2. train a single Gaussian (SG) to model each of the songs
3. compute distances between pairs of songs using the Kullback-Leibler divergence between respective SG models

We use a single Gaussian with full covariance per song [9] and calculate the acoustic similarity between two song models p and q as the symmetric KL divergence $D_{KL}(p,q)$

$$D_{KL}(p,q) = \frac{KL_N(p\|q) + KL_N(q\|p)}{2} \tag{1}$$

where

$$KL_N(p\|q) = 0.5 \log\left(\frac{\det(\Sigma_p)}{\det(\Sigma_q)}\right) + 0.5Tr\left(\Sigma_p^{-1}\Sigma_q\right) + 0.5\left(\mu_p - \mu_q\right)' \Sigma_p^{-1}\left(\mu_q - \mu_p\right) - \frac{d}{2} \tag{2}$$

$Tr(M)$ denotes the trace of the matrix M, $Tr(M) = \Sigma_{i=1..n} m_{i,i}$.

Because we tried to keep memory requirements low, we decided not to store the full distance matrix, but compute song-to-song similarities online. In the current implementation, the calculation of a full distance matrix row for 9500 songs takes around 50ms on a standard PC-based system.

3.4 Map generation

For the 3D visualization, we refined the main ideas from *Islands of Music* [11] and *Neptune* [5], implemented as an interactive 3D virtual landscape walkthrough. Central to this type of visualization is a terrain heightmap, which is generated automatically from the database of available tracks. The visualization uses an islands metaphor, where islands represent regions of similar-sounding music.

The heightmap profile was derived from a 2D MDS [2] solution that approximates distances derived from a combination of acoustic and metadata similarity between tracks as closely as possible. We chose not to use Self Organizing Maps because they inherently rely on an embedding of data points in a high dimensional vector space, which clearly is not the case when dealing with pure proximity data. One common way to construct vector spaces from proximity data – interpreting the distances of all data points to a data (sub)set – did not seem feasible because the database changes continuously (per day, on average 15 tracks are added, while others are deleted), and we needed a way to integrate new data without a complete recalculation of the map.

Although calculating an MDS solution from pure audio similarity tends to preserve distances as well as the local neighborhood of songs, it did not not produce visually discriminable clusters in 2D space. Each artist in the Soundpark carries two genre labels, which he or she can select

[4] http://www.cherrypy.org

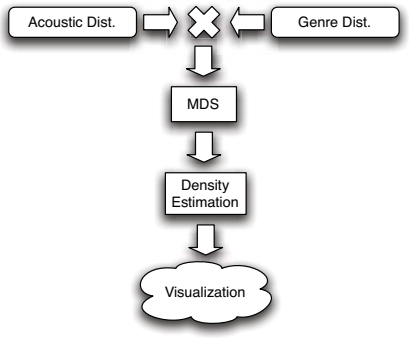


Figure 1. Map generation&visualization

freely from a set of 6 (Electronica, Funk/Soul, HipHop, Pop, Rock, Reggae). The desirable result from the map generation procedure was a layout where (1) tracks with equal genre labels are placed close to each other (2) the distances between tracks with equal genre labels still reflect mutual audio distances. Therefore, we decided to modify the audio similarity space by constructing a combined distance measure from the audio distance $\mathcal{D}_a$ and the genre distance $\mathcal{D}_g$:

$$
\begin{aligned}
\mathcal{D}_a(t_1,t_2) &= \overline{\mathcal{D}_{KL}} + w_a \cdot (\mathcal{D}_{KL}(t_1,t_2) - \overline{\mathcal{D}_{KL}}) \\
\mathcal{D}_{scaled}(t_1,t_2) &= 1 - \exp(-D_a(t_1,t_2)/100)) \\
\mathcal{G}_k(t_i) &= 1 \text{ iff } t_i \text{ has genre k, 0 otherwise} \\
\mathbf{G}(t_i) &= (\mathcal{G}_0(t_i)...\mathcal{G}_5(t_i)) \\
\mathcal{D}_g(t_1,t_2) &= 1 - \frac{\mathbf{G}(t_1)\cdot\mathbf{G}(t_2)}{min(\Sigma_k(\mathbf{G}_k(t_1)),\Sigma_k(\mathbf{G}_k(t_2)))} \\
\mathcal{D}_c(t_1,t_2) &= \mathcal{D}_{scaled}(t_1,t_2)\cdot((1-w_g)+w_g\cdot\mathcal{D}_g(t_1,t_2)) \\
0 \leq\ & w\ < 1
\end{aligned}
$$

$\overline{\mathcal{D}_{KL}}$ denotes the average audio distance in the data set. The weight factors w_a and w_g determine the influence of the tracks' audio distance and artists genre distance, respectively, on the overall distance measure. We chose values $w_a = 0.6, w_g = 0.9$ in order to enforce strong discrimination between tracks with no genre overlap. The rescaling step to calculate $\mathcal{D}_{scaled}(t_1,t_2)$ was inspired by [12].

Combined track-to-track distances are fed into the MDS module, which iteratively calculates a 2D layout, whereupon 2D positions are determined such that their respective distances approximate the original distances as closely as possible. We used Chalmers' [1] optimized spring model-based implementation. The complexity of the force calculation in this algorithm are reduced to $O(1)$ (by stochastically sampling the dataset and incrementally building a nearest-neighbor set for each point in the MDS problem by keeping the V nearest neighbors to each point over iterations), therefore its overall complexity is in $O(N^2)$, which is a crucial factor when dealing with data sets in the order of 10000 items and more (it should be noted that we found it necessary to increase the size of the sampling set to 20 in order to avoid falling into local minima during the solution of the MDS task, see [1] for a detailed discussion of the peculiarities of the algorithm). After $2N$ iterations (where N is the number of music tracks) the layout was assumed to be stable and the calculation was aborted.

From the low-dimensional track positions, we calculated a heightmap profile by applying a kernel density estimation [3] algorithm to the 2D point cloud, interpreting the estimated densities at points (x_i, y_i) as height values. The heightmap profiles are written to a binary file, which is then made available to the visualization client via a web server.

Because it is not feasible to execute the MDS procedure on each track import (which happens several times per day and requires the extraction of a full distance matrix), we place a newly imported track by calculating a weighted average position from its 5 nearest neighbors and do the MDS recalculation only during low-traffic times.

4 USER INTERFACE

Currently, two user interfaces to the recommendation system have been implemented: (1) A more traditional, Adobe Flash-based MP3 player interface with a small integrated visualization of similar tracks to the currently played one (see 4.1) and a downloadable, fully interactive, 3D visualization client (see 4.2).

4.1 Web player

The web player can be launched from within an artist's web page on the Soundpark website by clicking on one of the artist's tracks. Additionally to offering the usual player interface (start, stop, skipping forward/backward) it shows similar songs to the currently playing one in a text list and in a graph-based visualization (see figure 2).

The graph visualization displays an incrementally constructed nearest neighbor graph (number of nearest neighbors = 5), whereas nodes having an edge distance greater than two from the central node are blinded out. Figure 3 demonstrates the dynamic behavior of visualization (to simplify things, we have chosen a nearest neighbor number of 3 for this sketch): (1) User clicks on a track, visualizaton displays track (red) and the 3 nearest neighbors (green), (2) user selects track 4 by clicking on it, the visualization shows the track and its 3 nearest neighbors; note that track 2 – who is amongst the nearest neighbors to track 1 – is also in the NN set of track 4. (3) user selects track 5 as the new center, track 1 – which was nearest neighbor to track 4 in the preceding step – is also nearest neighbor to track 5. In the

Figure 2. Player and SoundGraph visualization

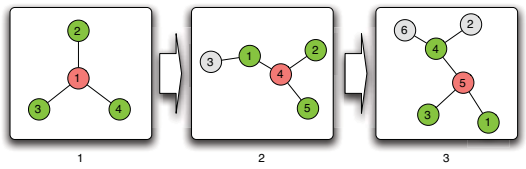


Figure 3. Interaction sequence in SoundGraph

long run, the re-occurrence of tracks in the NN sets indicates the existence of several connected components in the nearest neighbor graph.

4.2 3D visualization

The 3D visualization was implemented as a cross platform Java WebStart [5] application. The decision against a direct integration with the webpage had several reasons: (1) Flash cannot take full advantage of accelerated graphics hardware, which is ubiquitous nowadays, (2) Flash and Java applets both cannot access the local harddrive without quirks, which was necessary for local data caching.

The application was implemented using the JMonkey Engine [6] 3D framework. It implements an animated walkthrough of the island landscape; the user is put in a first-person perspective, and she can control the application by the well-known WASD+mouse method (W/A/S/D keys for movements, mouse for controlling the viewing/walking direction). Tracks are represented as images that are positioned at their appropriate positions and heights. By focussing a track image and clicking on it with the mouse, the user can start playback of a track. Figure 4 shows a screenshot of the running application.

[5] http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp
[6] http://www.jmonkeyengine.com/

Figure 4. The 3D application

By using optimized rendering algorithms (space subdivision with quadtrees), we were able to maintain interactive framerates (∼35 frames per second), even for large scenes consisting of ∼9500 tracks.

5 EVALUATION

5.1 Graph theoretical considerations

Our analysis of the incrementally constructed nearest neighbor graphs concentrates on how likely it is that individual songs are reached when users browse through the graph. The analysis is done on an evaluation data base of 7665 songs. With the number of nearest neighbours nn equal 3 (5), 56.79% (65.28%) of the songs can be reached in principle. The other songs are never part of any of the nearest neighbour lists. Next we constructed the full nearest neighbour graphs emanating from all of the songs by incrementally expanding all subgraphs. As soon as an already visited song is reached again, the corresponding subgraph is fully constructed. The size of the full nearest neighbour subgraphs ($nn = 3$) for 96.20% of the songs is between 597 and 957, for the remaining 3.8% it is only between 4 and 46. For $nn = 5$ there is a similar behavior with more songs being reached: the size of the full nearest neighbour subgraphs for 97.91% of the songs is between 2306 and 2669, for the remaining 2.09% it is only between 6 and 50. Looking at the strongly connected graphs that exists in our data set helps to explain this surprising behavior. For our incrementally constructed nearest neighbor graph, a strongly connected component (SCC) is a subgraph where every song is connected to all other songs traveling along the nearest neighbour connections. Using Tarjan's algorithm [13] to find all SCC-graphs in our nearest neighbour graph with $nn = 3$

($nn = 5$), we find that there is one single large SCC with the size of 543 (2231) songs. The remaining 5913 (4674) SCC have a size of only 1.2 (1.16) on average. All our results seem to indicate, that there exists one large tightly connected subgraph that all other songs lead to when travelling along the nearest neighbour connections.

5.2 Quality of map visualization

To quantify the success of mapping the high dimensional distance space to the low dimensional map visualization, we computed the following measures of neighborhood preservation. We obtained an MDS solution D_a^{MDS} using only the audio distances $\mathcal{D}_a$ as input. We obtained another MDS solution D_c^{MDS} using the combined distances $\mathcal{D}_c$ as input. For every song, we compute sets of n nearest neighbours N_a, N_a^{MDS} and N_c^{MDS} using $\mathcal{D}_a$, D_a^{MDS} and D_c^{MDS} as distance measure. We also compute the set of n nearest neighbours Ng_a within genres using $\mathcal{D}_a$ as distance measure, i.e. only songs with identical genres are allowed to be part of the nearest neighbour set. We next compute the percentage of common (overlapping) nearest neighbours $O(N_a, N_a^{MDS})$ between neighbour sets N_a and N_a^{MDS}, as well as $O(Ng_a, N_c^{MDS})$ between Ng_a and N_c^{MDS}. Whereas $O(N_a, N_a^{MDS})$ measures the neighbourhood preservation for a mapping based on audio alone, $O(Ng_a, N_c^{MDS})$ measures how well local audio similarities within genres are preserved in mappings based on the combined distances. This is done for a random subset of 1000 songs with varying size of n and the average overlaps are depicted in Fig. 5. As can be seen, the preservation of local neighbourhoods within genres in the combined map visualization is even better than the preservation of neighbourhoods based on audio information only. Combining audio and genre information not only allows for more interesting map designs, but also respects audio similarity at a local level.

5.3 Usage statistics

In this section, we will present analyses that prove the acceptance of music recommendation technology in everyday use. Note that we did not evaluate usage behavior of the 3D visualization yet, as this part of the system was very recently finished.

By analyzing webserver log files, we were able to verify the following hypotheses about how it will be possible to change the music consumption behaviour of FM4 Soundpark users by utilizing music recommendation technology:
(1) The new technology is used by a significant number of users
(2) while the absolute number of track accesses might stay constant, the number of *distinct* track accesses increases,
(3) track accesses are more evenly distributed across the entire track catalogue, that is, the age distribution of downloaded tracks is becoming flatter.

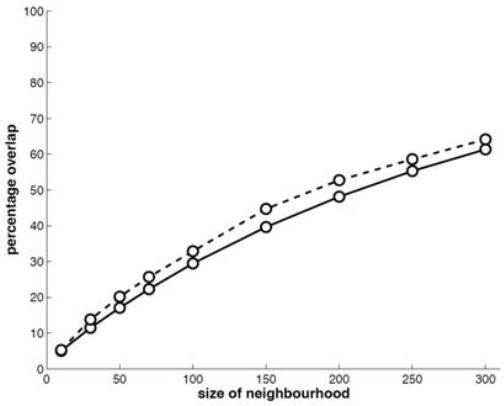


Figure 5. Percentage of overlap (y-axis) of neighbourhood sets $O(N_a, N_a^{MDS})$ (solid) and $O(Ng_a, N_c^{MDS})$ (dashed) for varying size of neighbourhood (x-axis).

Figure 6(a) plots the number of distinct tracks that have been downloaded per day between 2008-03-23 and 2009-03-29. Our recommendation service was launched on 2008-05-06. The peak at this date is clearly visible, and although the numbers turn down again during the following days, they clearly remain at a higher average level than they were before. The distinct track access numbers before/after the launch were distributed according to

	min	median	mean	max	stddev
before	17	338	359	781	137
after	41	593	672	4355	394

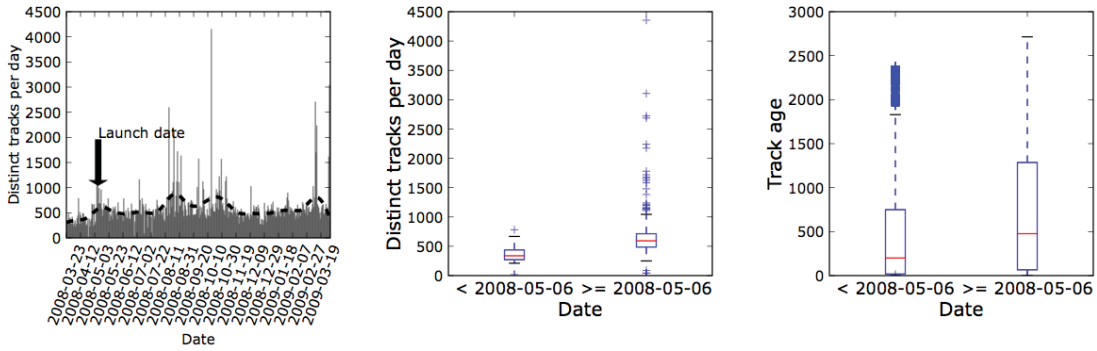
The ages of accessed tracks in days were distributed according to

	min	median	mean	max	stddev
before	0	200	483	2383	613
after	0	476	766	2714	795

The boxplots in fig. 6(b) and fig. 6(c) give a better visual impression of the effect of the recommendation service.

6 CONCLUSIONS

We have presented a real-life implementation of a music recommendation system that incorporates (1) purely content-based recommendations based on a seed track, (2) a 2D visualization based on pure audio similarity, and (3) an interactive 3D visualization based on a combined (audio and metadata-based) distance measure. We showed that recommendations based on a k nearest neighbor approach are



(a) (b) (c)

Figure 6. Usage statistics

likely to run into cycles if k is too small. We have presented a combined audio-and-metadata distance measure, whereas audio- and metadata-contributions can be weighted as required, and we have shown that a 2D MDS projection of a data set based on this measure respects audio similarity on a local level, while the coarse structure reflects distances calculated from the metadata. To check the usefulness of our system, we analyzed nearest-neighbor graphs calculated from pure audio similarity with graph theoretical methods, we analyzed MDS solutions with respect to neighborhood- and distance-preserving properties, and we performed statistical analysis of web server logfiles to analyze usage behavior. The results indicate that the approach we chose works reasonably well for our specific problem area.

7 REFERENCES

[1] Matthew Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *Proceedings of IEEE Visualization*, 1996.

[2] M.F. Cox and M.A.A Cox. *Multidimensional Scaling*. Chapman and Hall, 2001.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[4] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[5] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard 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.

[6] Paul Lamere and Douglas Eck. Using 3d visualizations to explore and discover music. In *Proceedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria, September 2007.

[7] Stefan Leitich and Martin Topf. Globe of music - music library visualization using geosom. In *Proceedings of the 8th International Conference on Music Information Retrieval*, 2007.

[8] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the 1st International Conference on Music Information Retrieval*, Plymouth, Massachusetts, 2000.

[9] Michael Mandel and Dan Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval*, London, United Kingdom, 2005.

[10] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. Playsom and pocketsomplayer: Alternative interfaces to large music collections. In *Proceedings of the Sixth International Conference on Music Information Retrieval*, 2005.

[11] Elias Pampalk. Islands of music: Analysis, organization, and visualization of music archives. Master's thesis, Vienna University of Technology, 2001.

[12] Elias Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, March 2006.

[13] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.