Personalized Music Recommendation in a Mobile Environment

Claus Schabetsberger
Department of Computational Perception
Johannes Kepler University Linz, Austria
claus.schabetsberger@gmail.com

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

ABSTRACT

Addressing the spiraling amount of music and video consumption via streaming services, in particular on mobile devices, we present a music player application for the Android platform, which employs a hybrid approach to generate a list of track recommendations for a user. We propose and evaluate two different algorithms, namely a content-based algorithm and an approach that exploits social similarity. While the former is based on rhythm features, the latter exploits "related videos" relations from `YouTube`. We show via a user questionnaire that recommendation results based on content slightly, but statistically significantly, outperform the social approach. Given that full audio content is not available immediately in a streaming environment, however, we suggest a hybrid, dynamic approach to music recommendation.

Playlists are created as a linear, user-adjustable mixture of both content and social similarity. They are offered to the user via an Android application dubbed "Beat Commander". Besides displaying the results of the playlist generation approach as text, the player features a dynamic visualization of the playlist, using a version of Sammon's mapping.

Keywords

mobile music player, hybrid music recommendation

Categories and Subject Descriptors

Information systems [**Information search and retrieval**]: Music recommendation

1. MOTIVATION

In the past few years, we have witnessed a considerable shift in how people consume video and music. The emergence of streaming services such as `YouTube`[1] and `Spotify`[2] has shown the need for novel techniques to sift through the enormous amounts of multimedia data available at the user's fingertips. Also the devices we use to consume multimedia material have changed from stationary PCs to mobile devices such as smart phones. Addressing these developments, we propose a novel, hybrid approach to recommend music in a mobile environment. Our approach focuses on `Android`[3]-based smart phones, where the minimum required version is 4.1 Jelly Bean. It is based on music features extracted from the audio content directly on the device and on contextual, social features inferred from `YouTube` data.

[1] `http://www.youtube.com`
[2] `http://www.spotify.com`

MoMM2013, 2-4 December, 2013, Vienna, Austria.

2. RELATED WORK

Different hybrid music recommendation approaches have been developed, but only a few of them are designed for a mobile environment. Among them, Cai et al. [1] propose a hybrid approach in which social information from `Last.fm`[4] and `Pandora`[5] is combined with music content features to train a support vector machine (SVM) [11]. This SVM is then used to generate a list of recommended tracks. However, this solution has the drawback that the SVM has to be retrained whenever new songs are added to the music collection. Additionally, this work is not intended to be used in a mobile environment.

Wang et al. [12] propose a system based on audio content, which tries to determine the user activity as the context by using sensors in a conventional smart phone. The approach then tries to find similar audio tracks according to the audio content determined by autotagging and implicit user feedback. If, for instance, a user listens to a song only a few seconds or until the end, this feedback is used to make recommendations. Depending on the information about the user's context and the implicit feedback, a probabilistic model is used to find most similar tracks. Since this model has to be trained offline, it would have to be re-trained over and over again every time when new tracks are added to the music collection.

Lee et al. [5] propose an approach similar to [12], but taking a pervasive computing perspective. Their idea is to build a mobile, context-aware application, which uses collaborative filtering to recommend music from users in similar situations. Additionally, they include non-standard smart phone sensors such as an electrocardiogram (EKG) sensor in order to capture the user context more precisely.

Yu et al. [14] present another hybrid recommendation approach designed for a mobile environment. At first, it determines whether a media item, such as an audio- or video file,

[3] `http://www.android.com/`
[4] `http://www.last.fm`
[5] `http://www.pandora.com`

conforms to the preferences of a user by comparing title, genre, actor and keywords between preferred media items and all entries in a database. After that, a Naive Bayes classifier is used to determine the probability for a media item to be recommended given a certain user situation. Finally, most similar media items are presented to a user in a particular way, by keeping in mind the output capabilities of the requesting smart phone device. Depending on these capabilities, a media item can be presented as video, audio or text. This approach, however, uses a prepared database that already contains extracted features of media items and it mainly focuses on recommendation rather than on feature extraction.

Pampalk et al. [7] suggest an application that relies only on audio content analysis to determine music tracks similar to a seed. By providing a skip button, a user is able to define music preferences. The latest track not skipped by the user is then used as input to content-based similarity models. Again, this work is not targeted at mobile usage.

3. MUSIC RECOMMENDATION

To generate a playlist based on extracted audio features as well as on social information, a web service was implemented, which has access to an internal database that holds audio features and user-related data such as favorite tracks. Additionally, the web service includes an interface to query `YouTube` and consequently compute social similarities.

A smart phone application based on Android was developed, which acts as an intelligent user interface to recommend and visualize tracks. While listening to music tracks, a user is able to rate them. These ratings are sent to the web service for further processing. Additionally, the application is responsible for extracting audio features, more precisely Fluctuation Patterns (FP) [6], which are also transmitted to the web service.

To generate a playlist, two different algorithms were implemented – one using content similarity defined on FPs, the other one by computing social similarity from information about related videos on `YouTube`. Both approaches require a seed track as input, which is a randomly selected favorite track of the user. The probability to select a particular seed track is given by its user rating.

Content-based Similarity

This approach uses FPs in order to find similar music tracks. FPs describe rhythm by modeling recurring beats. These features were proposed in [6], but are used in our work in a slightly different form, as described below. The entire audio feature extraction process is performed on the user's smart phone, while she is listening to music. This involves eight processing steps as shown in Figure 1.

When music is played on the smart phone, audio data in pulse code modulated format (PCM) is analyzed. To reduce the amount of input data and consequently speed up computation, the signal is first converted to mono. In order to avoid fade-in effects, the first 12 seconds are skipped. The remaining signal is split into 6-second-chunks, where only every third chunk is selected to be processed one at a time. Subsequently, the power spectrum is computed by applying a Fast Fourier Transform (FFT) on Hanning-windowed frames of 1024 samples each. In step 2, frequency values are grouped into critical bands according to the Bark scale. The purpose of steps 3 to 5 is to convert the sound pressure values given in the Power spectrum representation to Sone values, a perceptual measure of loudness. To obtain time-independent features, the modulation amplitude spectrum is calculated in step 6. This is done by applying a Discrete Fourier Transform (DFT) to each Bark band, yielding information on how often loudness changes per second. Since most loudness changes are in a range of [0,10] Hertz (Hz), only this range is considered. Step 7 employs another psychoacoustic transformation to weight loudness changes according to the model of Fluctuation Strength [3]. Depending on the frequency of loudness changes, the perceived intensity is different. Human perception of loudness is particularly sensitive at periodicities from 0 to 4 Hz, but then decreases slowly. Finally, a Gaussian blur filter is applied in order to make the resulting FP features more robust.

With this, one iteration of the feature extraction process is complete. During the analysis of a music track, the feature extraction process described is applied to 5 individual audio snippets of 6 seconds duration which corresponds to 90 seconds of audio data that is actually analyzed. This represents about a third of a conventional audio file of 5 minutes duration, but has also the benefit of not consuming too much energy from the smart phone of the user. Finally, the median of the resulting feature vectors is sent to the web service. The main difference to the originally proposed approach in [6] is that spectral masking is not applied, because this reduces too much of the lower frequencies of audio data as it turned out during experiments.

The content similarity algorithm on the web service uses FPs to return a list of similar tracks. First, the Euclidean distance between the seed track pattern p_i and all patterns in the database q_i are calculated. This value is normalized and the inverse of the distance is taken as similarity estimate, as shown in Equation 1. The value of q_{max} represents the maximum possible Euclidean distance between two patterns and there are in total n values in each of the compared FP feature vectors ($n = 1500$ in our implementation).

$$sim_c(p,q) = 1 - \sqrt{\frac{\sum_i^n (p_i - q_i)^2}{n \cdot q_{max}^2}} \quad (1)$$

One issue of the content similarity algorithm in a streaming environment is the cold start problem, i.e. similarity between tracks can only be computed if the audio is already available. The solution we propose here is twofold: (i) we start computing FPs "on the fly", when audio data comes in and (ii) we follow the idea of crowdsourcing, i.e., the more users are listening to a track the more likely the system already knows the respective FPs.

Determining previously unseen tracks and computing their FPs requires a second approach to select pieces similar to the ones already known by the system. This is due to the fact that audio content similarity can only be calculated on known tracks. In order to extend the set of pieces known to the user, however, another approach is needed. This is where the social similarity algorithm comes into play.

Social Similarity

The social similarity algorithm uses the "related videos" feature of `YouTube` to find tracks similar to a given seed. Roughly speaking, this feature works as follows: First the "keyword relevance" between videos is calculated using the title, summary, tags, and other information of `YouTube` videos. The

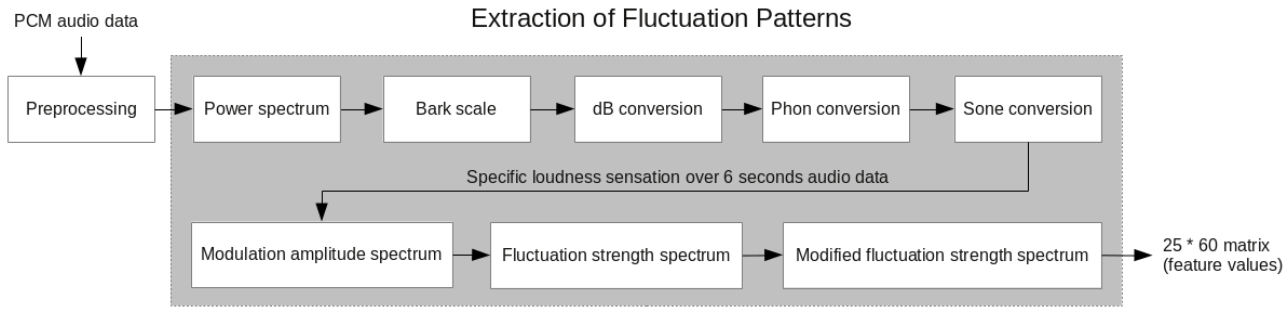

Figure 1: Extraction of Fluctuation Patterns

result of this is indirectly dependent on the taste of other users, because it returns only those videos as related, which share a popularity likewise to the selected seed track [13]. The popularity of a video will then be determined by measuring different signals, such as the click count of a video or the total duration of all views. [2] For example, if a user watches 50 seconds of a 1-minute-video, the popularity is assumed to be higher than if the user only watches its first 5 seconds.

In our approach to social similarity, we use `YouTube`'s related videos function to dynamically build a directed graph whose nodes are videos and all edges have uniform weight. An edge from video A to B is inserted if B occurs within the set of related videos of A. More sophisticated approaches are currently under investigation. Based on this graph, which is stored on the server and constantly extended while new related videos information comes in, we compute social similarities as follows: First the seed track marks the starting point in the related videos graph. The algorithm then tries to find N track with minimum distance to the seed track using Breadth-first search. Finally, the similarity between the seed track and the related track is determined as shown in Equation 2, where $d_{p,q}$ denominates the distance between the seed track and the related track.

$$sim_s(p,q) = \frac{1}{d_{p,q}+1} \tag{2}$$

Hybrid Music Recommendation

To determine a list of recommended tracks given a seed, the similarity scores produced by both algorithms are combined. Every recommended track is hence assigned a similarity vector, which contains a value for the content- and the social similarity. Tracks that were recently recommended are removed in order to avoid the same track being suggested over and over again, boring the user.

If an insufficient number of tracks are found at this point, for instance, because the system does not know about the user's favorites or the similarity scores for the user's selected tracks have not been computed yet, the list of recommended tracks is populated with tracks from the most popular video feed provided by `YouTube`. This frequently happens when a user starts the Android application for the first time.

Having a set of recommended tracks, the next step is to sort it according to the similarity to the selected seed track. For that, a linear combination of both kinds of similarity is computed according to Equation 3, which is eventually used to create a sorted list of recommended tracks, shown as playlist to the user. The weights w_c of the content similarity

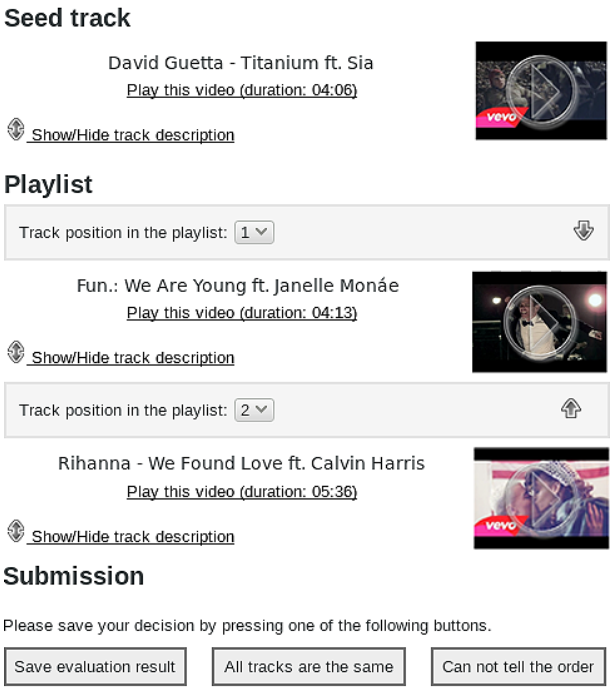

Figure 2: Example of an evaluation input form

algorithm and w_s of the social similarity algorithm can be defined by the user as described in Section 5.

$$sim(p,q) = sim_c(p,q) \cdot \frac{w_c}{w_c+w_s} + sim_s(p,q) \cdot \frac{w_s}{w_c+w_s} \tag{3}$$

4. EVALUATION

This evaluation aims at determining the precision of both similarity algorithms in terms of the relative track order. To this end, we use an A/B test, i.e., given a seed track and two target tracks, users are asked to tell which of the targets are more similar to the seed. This approach is similar to the one employed in the `TagATune` game [4]. An example input form is shown in Figure 2, which was provided to each participant together with detailed information about the intention of the evaluation. As both algorithms try to find most similar tracks for a particular seed track, it is expected that, for instance, the first recommended track is more similar to the seed track than the second recommended track.

In general, [10] has shown that current algorithms de-

Algorithm	Correct	Incorrect
Content similarity	82%	18%
Social similarity	63%	37%

Table 1: Overall evaluation results

pending on collaborative filtering produce better recommendations than content similarity algorithms. It is hence expected that our social similarity approach outperforms the proposed content-based method.

Experimental Setup

Evaluation was conducted using 50 tracks from 10 different genres (uniformly distributed) as seed tracks and 250 tracks with both types of similarity available as target set. 10 participants were asked to indicate for the 50 seed tracks which of the 2 target tracks are more similar to the seed. Participants were aged between 14 and 65 and had a different background: 3 of them were students, 2 were high-school graduates and 5 have passed secondary modern school. Only 1 participant was actually a musician.

The target tracks were selected using either content- or social similarity. Participants could also indicate that they were not able to tell which of the 2 targets were more similar to the seed. These cases were excluded from further processing. The precision was then computed using as ground truth these relative similarity judgments of the participants.

Evaluation Results

The evaluation results are summarized in Table 1. According to a χ^2-test [9], the content similarity algorithm produced more accurate playlists, with a significance level of 10%. In general, tracks recommended via content similarity were perceived more similar than tracks suggested by social similarity. This surprising result can be explained by: (i) the quite simple model of social similarity we employ, (ii) the related videos feature of `YouTube` not being tailored to the task, or (iii) the participants in the user study having a perception of music similarity that is better captured by the rhythm information reflected in the FP features.

We further evaluated precision of both approaches on the genre level. It turned out that both algorithms had most problems with country music. This can be explained by the fact that all recommended country tracks sounded very similar. The content similarity algorithm did not produce any error at all for the 5 genres Blues, Classic, Dance, Pop, and Rock, while the social similarity algorithm produced at least one incorrectly sorted playlist for each genre.

5. THE BEAT COMMANDER PLAYER

The Android application dubbed "Beat Commander" was designed to be user-friendly and simple. It communicates with the web service, for example, to submit FPs computed on the device, to request a playlist given the user's tracks, or to retrieve information about the URL of a video on `YouTube`. The application is also able to stream `YouTube` videos and to gather ratings from the user. To create playlists, the user can provide weights to indicate the influence of both recommendation algorithms on the playlist creation process.

The minimum hardware and software requirements of "Beat Commander" are summarized in table 2. The required hardware is comparable to a `Samsung Galaxy S`[6].

	Requirements
Hardware	1 GHz ARM Cortex-A8 processor 512 MB RAM 5 MB disc space
Software	Android 4.1 Jelly Bean

Table 2: Minimum application requirements

An important functionality of "Beat Commander" is to visualize the playlist, hence allow to navigate through tracks in an intuitive way. For that, different views are provided, some of which are shown in Figure 3. The first view illustrates music tracks as planets, which creates the impression that the user is sitting in a space ship watching the tracks move around him while navigating through a galaxy of music. A planet gets bigger when the respective music is played and it may disappear when the user switches to the next music track, thus enforcing a novel list of recommended tracks. Coordinates of planets are calculated by the web service by applying Sammon's mapping [8] on the full similarity matrices of the tracks in the recommended set and the seed track. The second view shows the playlist in a more structured way, where tracks are listed in order of their similarity to the seed. This view also reveals more detailed information, in particular the similarity of the recommended tracks to the seed. Additionally "Beat Commander" provides a view of the `YouTube` video and visualizations of the audio content analysis steps presented in Section 3.

6. CONCLUSIONS AND FUTURE WORK

We presented two approaches to define music similarity based on audio content and based on social information inferred from `YouTube`. We integrated the two methods into a mobile application that uses the resulting similarities to automatically create lists of recommended tracks matching the user's taste. Furthermore, we evaluated the quality of playlists produced by the content- and the social approach via a user study involving A/B listening tests and we showed the content-based approach outperforms the social approach in this setting. Nevertheless, the combination of the two is necessary as content data is not available from the beginning in a streaming environment.

Future work will look into more elaborate methods to incorporate music context and social information into the system. Furthermore, we plan to integrate different methods to select seed tracks and to include a more comprehensive user model, taking into account environmental data readily available on mobile devices, for example, location, weather, activity, and time.

7. ACKNOWLEDGMENTS

This research is supported by the Austrian Science Fund (FWF): P22856, P25655, and the EU FP7 project PHENICX: 601166.

8. REFERENCES

[1] J. Cai, J. Francis, and S. Gheysens. Creating a Hybrid Music Recommendation System from Content and

[6] `http://www.samsung.com`

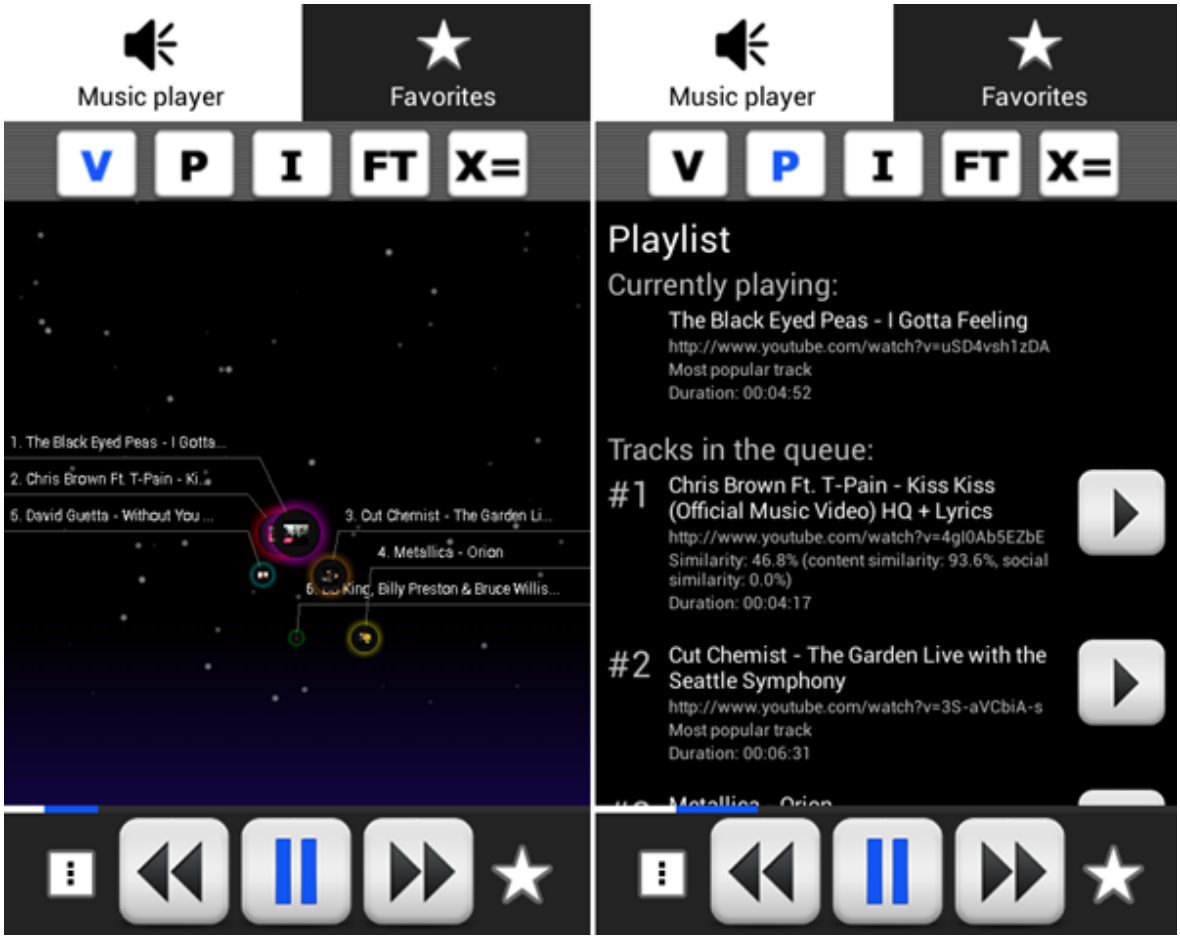

Figure 3: Two selected views of the music player

Social-Based Algorithms. *Governor's School of Engineering and Technology Research Journal*, 2009.

[2] J. Doe. Changes to Related and Recommended Videos, March 09 2012. `http://youtubecreator.blogspot.co.at/2012/03/changes-to-related-and-recommended.html`.

[3] H. Fastl. Fluctuation Strength and Temporal Masking Patterns of Amplitude-Modulated Broad-Band Noise. *Hearing Research*, 8:59–69, 1982.

[4] E. Law and L. von Ahn. Input-Agreement: A New Mechanism for Collecting Data Using Human Computation Games. In *27th International Conference on Human Factors in Computing Systems*, Boston, USA, 2009.

[5] H. Lee and J. Kwon. Situation and Social Awareness-based Personalized Recommendation Service in Pervasive Computing Environment. In *5th International Workshop on Smart Environments and Ambient Intelligence*, San Diego, USA, March 2013.

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

[7] E. Pampalk, T. Pohle, and G. Widmer. Dynamic Playlist Generation Based On Skipping Behavior. In *6th International Conference on Music Information Retrieval*, London, UK, September 2004.

[8] J. W. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.

[9] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 3rd edition, 2004.

[10] M. Slaney. Web-Scale Multimedia Analysis: Does Content Matter? *IEEE MultiMedia*, 18(2):12–15, 2011.

[11] V. N. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, UK, 1998.

[12] X. Wang, D. Rosenblum, and Y. Wang. Context-Aware Mobile Music Recommendation for Daily Activities. In *20th ACM International Conference on Multimedia*, Nara, Japan, October–November 2012.

[13] S. Wittens. Six Degrees of YouTube, August 10 2012. `http://www.strutta.com/resources/posts/six-degrees-of-youtube`.

[14] Z. Yu, X. Zhou, D. Zhang, C.-Y. Chin, and X. Wang. Supporting Context-Aware Media Recommendations for Smart Phones. *IEEE Pervasive Computing*, 5(3):68–75, July-September 2006.