Beyond "Hitting the Hits" – Generating Coherent Music Playlist Continuations with the Right Tracks

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Lukas Lerche
TU Dortmund, Germany
lukas.lerche@tu-dortmund.de

Iman Kamehkhosh
TU Dortmund, Germany
iman.kamehkhosh@tu-dortmund.de

ABSTRACT

Automated playlist generation is a special form of music recommendation and a common feature of digital music playing applications. A particular challenge of the task is that the recommended items should not only match the general listener's preference but should also be coherent with the most recently played tracks. In this work, we propose a novel algorithmic approach and optimization scheme to generate playlist continuations that address these requirements. In our approach, we first use collections of shared music playlists, music metadata, and user preferences to select suitable tracks with high accuracy. Next, we apply a generic re-ranking optimization scheme to generate playlist continuations that match the characteristics of the last played tracks. An empirical evaluation on three collections of shared playlists shows that the combination of different input signals helps to achieve high accuracy during track selection and that the re-ranking technique can both help to balance different quality optimization goals and to further increase accuracy.

Keywords

Music Recommendation; Quality Factors; Evaluation

1. INTRODUCTION

The automated creation of music playlists – a special form of music recommendation – is a typical feature of modern digital music playing applications, and playlist generators are nowadays used both by online music platforms and by "offline" music applications that, e.g., run on smartphones. Over the last fifteen years, a variety of playlist generation approaches has been proposed in the literature [4, 12]. These next-track recommendation approaches differ from each other, e.g., with respect to the kinds of inputs they expect, which forms of additional data about the tracks they can process, or which quality criterion they seek to optimize.

A distinctive feature of music playlist generation is that the recommended items are immediately "consumed", usually in the specific order determined by the playlister. The problem is therefore not only to determine tracks that appeal to the general taste of the listener, but to generate playlists that obey additional constraints, e.g., with respect to the homogeneity of artist, genre, or tempo or the smoothness of track transitions [2, 13, 19, 21]. Furthermore, there are applications in which the generation of virtually endless playlists is required (radios). In such cases, the recommendations should not only be coherent in themselves but also match the most recently listened tracks.

In this work, we focus on such immediate next-track recommendations given a history of recent tracks. We apply a two-phase approach. In the first phase we determine a set of tracks that generally seem suitable to be combined with the most recent listening history. We base the selection and ranking of the tracks on a multi-faceted scoring method which combines track co-occurrence patterns in publicly shared playlists, music and meta-data features as well as personal user preferences. Similar to the approaches of [3, 4, 8, 14, 15], we aim to optimize the track selection accuracy in this phase. As a basis for the optimization, we use pools of music playlists that were manually created and shared by music enthusiasts. We use the track hit rate to assess to which extent the algorithms are capable of selecting tracks that were also chosen by the users.

In the second phase, we optimize the set of the immediate next tracks to be played. Optimizing for accuracy alone can be insufficient as the inclusion of a few wrong tracks can be easily detrimental to the perceived service quality as a whole [6]. The main idea is therefore to re-rank the tracks selected in the previous phase in a way that the resulting playlist continuation matches the characteristics of the recent history. Specifically, we use a generic optimization procedure that *minimizes the difference* between the history and the continuation in terms of one or more desired quality dimensions. Thereby, we avoid the definition of global quality levels to be achieved as these levels can depend on the "goals" of the playlist. Including, e.g., mostly popular tracks can be favorable for a party playlist. If the goal, in contrast, is track discovery, genre homogeneity and the inclusion of less popular tracks could be advisable.

To evaluate the optimization procedure, we compare the accuracy-optimized playlists from the first phase with the continuations obtained after the second phase in different dimensions. Specifically, we first evaluate to which extent the re-ranked continuations lists are more similar and thus coherent with the history. In addition, we determine the impact of the re-ranking process on track selection accuracy.

RecSys'15, September 16–20, 2015, Vienna, Austria.

http://dx.doi.org/10.1145/2645710.2645755.

2. FACETED TRACK SCORING

In this section, we present the details of our multi-faceted weighted scoring approach. The goal is to determine a relevance score for each possible next track given a playlist history h (sequence of tracks) using different input signals.

2.1 Baseline Scoring Approach

In [4], a number of playlisting algorithms based on patterns in publicly shared playlists were benchmarked. The results showed that a k-Nearest-Neighbor-based (kNN) approach worked best in terms of accuracy across all datasets in particular for the shorter list lengths that are relevant in our context. As we are interested in high-quality next-track recommendations, we will use kNN as a baseline. Given a history h and a set N_h of nearest neighbor playlists of h, we compute the kNN score of a target track t^* with $sim_{cosine}(h, n)$ as the binary cosine similarity of track occurrences in h and n. $1_n(t^*) = 1$ if n contains t^* and 0 otherwise.

$$score_{kNN}(h, t^*) = \sum_{n \in N_h} sim_{cosine}(h, n) \cdot 1_n(t^*) \qquad (1)$$

kNN was also used as a baseline in [8]. Small hit rate increases were achieved at long list lengths by combining it with tag-based track information. The work also showed that the kNN method performs better than recent methods like BPR [17] for the given task. In [8] however, a very small value for $k = 10$ was used. Since our experiments show that larger values for k ($k = 300$) lead to significantly higher hit rates and to the best results, we will not use the tag-enhanced method of [8] as a baseline.

2.2 Computing Additional Suitability Scores

Playlist creators can have certain "themes", goals or quality criteria in mind when they design a playlist. The idea of our faceted scoring method is to combine the kNN approach with additional suitability scores. If we, for example, detect that the homogeneity of the tempo of the tracks is most probably a guiding quality criterion, we should give an extra relevance weight to tracks that are similar in tempo to those in the history; if we observe that several tracks in the history were annotated by users with certain tags, we should increase the relevance score of tracks with similar tags.

Generally, the combination of different scores shall serve two purposes: (1) increasing the hit rate as more relevant tracks receive a higher aggregated score and (2) making the playlist continuation more homogeneous, which is often a desirable goal in the Music Information Retrieval literature.

In our experiments, we tested the following scores as examples to validate our approach. The selection was based on the availability of track metadata in public sources.[1]

2.2.1 Measuring "Content-based" Similarity

There are several ways of determining the similarity of music tracks, e.g., by modeling the distribution of track features [2]. In our work, we retrieved the social tags[2] that were assigned to tracks by users on Last.fm as they are valuable indicators of what a certain playlist is about [8]. We removed irrelevant tags like "my favorite" or "like it", retained the 1,000 most frequent tags and computed TF-IDF (Term Frequency/Inverse Document Frequency) vectors for each track[3]. Given a history h consisting of tracks $t_i, \ldots, t_n$ with the TF-IDF vectors t_1^v to t_n^v, we compute the score of a target track t^* using its TF-IDF vector t^{*v} as follows.

$$score_{content}(h, t^*) = sim_{cosine}\left(\frac{\Sigma_{t_i^v \in h} t_i^v}{|h|}, t^{*v}\right) \qquad (2)$$

2.2.2 Matching Numerical Features

Numerical track features can be music-related like the tempo or loudness or describe other aspects like the release year or track's general popularity. Such features can be relevant factors which determine the selection of tracks, e.g., for playlists like "Hits of the 90s", or "Best of" mixes.

The assumption of our scoring scheme for numerical values is that if a feature was actually relevant for the selection of tracks, the spread and variance of the values will be low. Given a history h and a feature value f_{ti} for a track t_i in the history, we therefore first compute the mean μ and standard deviation σ of the observed values. If there are too few values available – some might be unknown – or the standard deviation exceeds some threshold δ, the score is 0.

Otherwise, given the feature value f_t^* for a target track t^* we use the value of the probability density function of a Gaussian distribution as a score, where μ and σ are computed based on the value distributions in the history:

$$score_{numfeature}(h, t^*) = \frac{1}{\sigma_h \sqrt{2\pi}} e^{-\frac{(f_t^* - \mu_j)^2}{2\sigma_h^2}} \qquad (3)$$

If the standard deviation is 0, e.g., because all tracks were released in the same year, we return a specific maximum score. In case the feature value f^* is too far away from the mean (e.g., more than 2 standard deviations), the score is again set to 0. The choice of the maximum score (after normalization) and the tolerance regarding the deviation from the mean can be made depending on the data.

2.2.3 Considering Personal Preferences

Personal preferences, e.g., regarding favorite artists, can influence what people add to their playlists. In our experiments we therefore also include an example of a personalized scoring scheme. We base the method on content features as only a limited number of playlists is available per user.

To build a "long-term" content-based profile for user u with the recent listening history h, we first compute an averaged TF-IDF vector CP_h from all playlists created by u. For a track t^* with a corresponding TF-IDF vector t^{*v}, the personalized score ($score_{contentpers}$) is then based on a weighted combination of the similarity of t^* with the current history $score_{content}(h, t^*)$ and the similarity of t^{*v} with the long-term profile CP_h, i.e.

$$score_{contentpers}(h, t^*) = \alpha \cdot score_{content}(h, t^*) + (1 - \alpha) \cdot sim_{cosine}(CP_h, t^{*v}) \qquad (4)$$

Other combination types are again possible, e.g., using a decay factor for older playlists, in case the playlist creation dates are available.

[1] Understanding the true theme of a playlist is in general challenging as it might be based on data like track lyrics or instrumentation aspects, which are not publicly available for most tracks.

[2] With "content", we refer to social tags and not to the audio signal, as is sometimes done in the field of Music Information Retrieval.

[3] Additional experiments using Probabilistic Latent Semantic Analysis (pLSA) for topic detection did not lead to better results.

2.3 Combining the Scores

Depending on the available data, various combinations of scores of different types can be used. In our experiments, we tested different configurations using a weighted scoring scheme. Given a set of scoring functions S, the score is

$$score_{overall}(h, t^*) = \Sigma_{score_i \in S}\ score_i(h, t^*) \cdot w_i \quad (5)$$

where w_i is a configurable weight term for each $score_i \in S$.

The selection of the scores as well as their corresponding weights depend on the goals that should be achieved. In this work, we systematically tested different values for the weight of each score and selected the best ones (listed in Table 2) with respect to accuracy.

2.4 Treatment of Single-Artist Histories

In particular on Last.fm, we can find playlists that contain tracks of only one single artist. If a history h has only one artist A, we apply an additional heuristic and compute a special artist score (AS) for tracks by A. In the experiments, we used a baseline AS value that is generally higher than the combined scores as computed in Equation 5. This ensures that the playlist continuation starts with the "right" artist. The tracks of artist A are then sorted according to their popularity, and we use a position-based decay function that assigns a lower AS value to less popular tracks of the artist and apply a minimum popularity threshold. The tracks of the other artists receive the usual combined score (Eq. 5).

3. NEXT-TRACK OPTIMIZATION

So far, we have designed a number of ways to score tracks with respect to a history of recent tracks. Given a specific dataset, our first goal is to find suitable weight factors and other parameters to optimize the prediction accuracy of the model, in our case in terms of the hit rate. Once this is done, our assumption is that the top-scoring tracks are generally well-suited for the given history.

The goal of the subsequent fine-tuning phase is to optimize the selection of the immediate next tracks. For that purpose, we will take a limited number of tracks from the top-scoring tracks and systematically re-rank them in order to better match one or more characteristics of the history. Since we limit the re-ranking to a smaller set at the top of the list, we (a) limit the computational complexity of the optimization process[4] and (b) can ensure that the potential accuracy losses are not too high.

3.1 Example and Design Rationale

Consider, as an example, that our optimization goal is to find a 10-track playlist continuation that maintains the tempo level of the recent history. In that situation, we pick, e.g., the top 30 tracks of the previously scored list and then systematically identify those 10 tracks among them which are most suitable in terms of the tempo. Note that the optimization goal here is *not* to find 10 tracks that all have the same tempo, but a set of tracks whose tempo distribution is as similar as possible to the current playlist. As another example, consider the diversity (of genres, artists, release years). Clearly, there is no global and context-independent optimum how diverse a playlist should be in general. In our approach, we therefore aim to find a set of tracks that mimics the features of the current playlist as good as possible and our optimization goal is always to *minimize the difference* between the characteristics of a feature in the playlist and the recommended immediate next tracks.

In some sense, this approach helps to identify a possibly underlying theme and corresponding track-selection rationale of the history and recreate it with the recommendations. If the tempo, for example, is consistently very slow in the history we can consider this as a possible design criterion and a similar continuation. If there is a strong variation in the tempo, this can be a possible design criterion as well and could be captured by our approach. Finally, our method can be applied in situations in which more than one factor might be relevant.

3.2 Quantifying Track List Characteristics

Technically, in order to optimize the next-track list to match the characteristics of a given history, we first have to define a way to measure the characteristics and their differences. We therefore introduce a function $\mathcal{F}_c(l)$ that quantifies the characteristic c for a list of tracks l. In this work, we distinguish three types of list characteristics.

Mean and Standard Deviation. For numerical characteristics such as the release year, the loudness or the tempo of tracks, a simple way of assessing their "distribution" over a track history is to use the mean and the standard deviation. $\mathcal{F}_c(l)$ is therefore defined by a set of two values

$$\mathcal{F}_c(l) = \{\mu_c(l), \sigma_c(l)\}$$

which can be combined in different ways to obtain a single numerical score.

Aggregate Measures. Some playlist characteristics can be assessed based on more complex aggregate measures. One example is to determine the diversity of a list based on the average of the pairwise similarities. In our experiments, we use the "Intra-List-Similarity" measure [22], i.e.,

$$\mathcal{F}_c(l) = \sum_{i \in l} \sum_{j \in l} \frac{sim_c(i,j)}{|l|^2}$$

To compute the value of $sim_c(i,j)$ in our experiments, we use the cosine similarity between the TF-IDF vector representations of the social tags for each track.

Mixture Models. Another possible approach is to fit Gaussian Mixture Models (GMM) to the distribution of the values in the playlist history and use the Akaike Information Criterion (AIC) to assess the difference between the history and the playlist continuation[5].

3.3 Optimization Procedure

Optimization goal. The generic optimization goal of the subsequently described procedure is to minimize the value of a function $d(\mathcal{F}_c(h), \mathcal{F}_c(r_n))$ which captures the difference between the characteristics of a given playlist history h and the top-n next-track recommendations r_n.

How the difference between the history h and r_n is actually computed, depends on the way the list characteristics are captured. In our experiments we use the absolute difference for the aggregate ILS measure, i.e., $|\mathcal{F}_c(h) - \mathcal{F}_c(r_n)|$ and the pairwise absolute difference for mean and standard deviation characteristics, i.e., $|\mu_c(h) - \mu_c(r_n)| + |\sigma_c(h) - \sigma_c(r_n)|$.

[4] Optimization-based playlisting approaches such as [16] can be computationally demanding due to the large number of available tracks.

[5] We ran experiments using GMMs, but do not report the results here as these more complex models for our datasets did not lead to better results with respect to the measures reported in the paper.

Algorithm 1: Top-n optimization algorithm. All possible swapping combinations between r_n and x_m are evaluated. $r_n^{i \leftrightarrow j}$ denotes a swap of track i in r_n with j.

input : **int** *maxSteps*; **Array** h, r_n, x_m
output: The updated top-n list r_n

for 1 **to** *maxSteps* **do**
  $\Delta_e \leftarrow 0$;
  $e_c \leftarrow d(\mathcal{F}_c(h), \mathcal{F}_c(r_n))$;
  for $i \in r_n$ *(backwards)*, $j \in x_m$ **do**
    $e'_c \leftarrow d(\mathcal{F}_c(h), \mathcal{F}_c(r_n^{i \leftrightarrow j}))$;
    if $e_c - e'_c > \Delta_e$ **then**
      $I_{best} \leftarrow i$; $J_{best} \leftarrow j$; $\Delta_e \leftarrow e_c - e'_c$;
  if $\Delta_e > 0$ **then**
    $r_n \leftarrow r_n^{I_{best} \leftrightarrow J_{best}}$
  else
    `break`;
return r_n;

Method. Let n be the length of the list of next tracks to be optimized. As a starting point, we take the first n recommendations of the list r produced by the faceted scoring method from Section 2 and determine the track list characteristics of the top-n list r_n as $\mathcal{F}_c(r_n)$. We then create an "Exchange List" x_m consisting of the m elements which immediately follow after r_n in r (e.g., tracks 11 to 30).

The optimization procedure shown in Algorithm 1 takes these lists – together with the history h – as an input and systematically exchanges elements from r_n with elements from x_m. In each iteration of the main loop, the best possible swap is determined. This is the one that leads to the largest reduction of e'_c. At the end, an exchange is only applied if it leads to a reduction of the difference between the characteristics of the history h and the updated top-n list. The algorithm stops when either no (measurable) improvement can be observed, all possible swaps are explored, or a maximum number of exchange steps were applied. The exact procedure is given in Algorithm 1.

Dealing with multiple goals. The general algorithm scheme can be used to optimize more than one goal at once, considering, e.g., both the tempo and the loudness in parallel. Instead of determining only one difference e_c, we would determine multiple difference values $e_{c1}, \ldots, e_{ck}$ and only perform a swap when the aggregated difference, e.g., the sum over all differences $e_{c1}, \ldots, e_{ck}$ (normalized to the same level), will be reduced. Weighted combinations of the individual error measures are possible as well.

4. EXPERIMENTAL EVALUATION

In the first part of the section, we analyze the effects of our faceted scoring method in terms of (a) accuracy, (b) homogeneity of the resulting lists, and (c) measure if the scoring methods help to generate list continuations that are coherent with the playlist histories. In the second part, we report the results of the re-ranking optimization procedure presented in the previous section. We use a significance level of $p = 0.05$ for all statistical tests throughout the section.

We used playlist collections from three different sources. One set of playlists was retrieved from Last.fm via their public API, one consists of playlists by music enthusiasts on the Art-of-the-Mix (AotM) platform published by [15], and one was shared with us by the 8tracks music platform[6].

Table 1: Statistics of the used datasets.

	Last.fm	AotM	8tracks
Playlists	2,978	1,040	6,714
Users	451	142	996
Avg. Playlists/User	6.60	7.32	6.74
Tracks	18,083	11,413	39,875
Avg. Tracks/Playlist	11.68	16.98	12.97
Avg. Track Usage	1.93	1.55	2.18
Artists	3,272	2,770	9,122
Avg. Artists/Playlist	4.55	12.76	12.06
Avg. Artist Usage	10.65	6.38	9.54
Avg. Genres/Playlist	16.83	39.18	38.06
Avg. Tags/Playlist	94.88	140.62	123.07

Table 2: Set of evaluated algorithms.

Alg.	**Description**			
PopRank	Ranks tracks according to their popularity, i.e., their occurrence in the training playlists			
CAGH, SAGH	Recommend the greatest hits of the artists in the playlist (SAGH) or the similar (collocated) artists (CAGH) [3]			
kNN300	k-Nearest-Neighbors with k=300			
Content	Similarity-based ranking (Eq. 2)			
Content-Pers	Sim.-based ranking with personalization. Last.fm, AotM: α=0.6, 8tracks: α=0.8 (Eq. 4)			
kNN300*	Weighted combination of kNN300 (w=1.0) with other scores (Eq. 5)			
	Weight for w on:	Last.fm	AotM	8tracks
	RY=release year	0.7	2.0	0.3
	TMP=tempo	0.7	2.0	0.3
	LD=loudness	0.5	0.5	0.3
	ART=single-artist heuristic	4.0	2.0	2.0
	CP=ContentPers.	0.3	0.3	0.3
	CPA=ContentPers and ART (w as above)			
	ALL=all available scores (RY, TMP, LD and ContentPers, w as above)			

Table 1 shows basic statistics and the following additional observations can be made. In the Last.fm dataset, a number of playlists contain tracks of only one or very few artists. On the other hand, having several tracks of one artist in a playlist is forbidden for public playlists on 8tracks and uncommon for data from AotM. For each playlist, we know a dataset-dependent user ID of its creator. From each user, we have at least 4 playlists, which allows us to apply the content-based personalization score. Furthermore, information about features like tempo, release year, or social tags was retrieved with the public APIs of Last.fm, theechonest.com, and musicbrainz.org. However, the metadata is only 75% complete on average across all tracks.

Table 2 shows the algorithms, hybrid variants and parameter settings that were used in the experiments.

[6]http://last.fm, http://artofthemix.org, http://8tracks.com

4.1 Results – Faceted Scoring

4.1.1 Accuracy

Method. We use the following four-fold cross-validation setup to measure accuracy [3, 4, 8]: The available playlists are split into training and test sets. As usual, the algorithms learn their models on the training sets. From each playlist in the test set, we hide the last track (resulting in a history h) and let the algorithms recommend playlist continuations, i.e., predict the hidden track. A "hit" is registered whenever the generated top-n list contains the hidden element. Therefore, the hit rate is the fraction of playlists in the test set for which the hidden track was found. Beside the hit rate, we report the Mean Reciprocal Rank (MRR) measure, which takes the position of the hit into account[7].

Results. Table 3 shows hit rate and MRR at list length 100. Determining the accuracy for this list length gives us flexibility in setting the size of the Exchange Set later on. Note in addition that top-n lists of this and even much larger sizes are common in this domain [3, 4, 8]. The lowest and highest values are marked in light gray (lowest) and a bold font and a dark gray background (highest) respectively.

As expected, kNN300 has a competitive performance compared with the other baselines and recommending the most popular tracks to everyone can sometimes be a better strategy than using noisy tags alone (Content). The highest accuracy[8] across all three datasets can be achieved by combining kNN with the *personalized* content-based algorithm from Section 2.2.3 and the artist heuristic from Section 2.4 (kNN300CPA). The differences between the best performing hybrid method and the kNN baseline method are statistically significant on both measures and all three datasets. The results therefore indicate that different signals (track usage patterns, social annotations and personal long-term preferences) in combination with specific heuristics should be used to obtain high accuracy. While kNN300CPA consistently works best in absolute numbers, the differences to the second-best method for individual datasets and measures are not always significant.

Combining kNN with the numerical features from Section 2.2.2 (release year, tempo and loudness) can in some cases lead to modest accuracy improvements. On AotM, playlist creators seem to pay attention to aspects like a consistent loudness level in the playlists, which explains the improvement of kNN300LD over kNN300. For the other datasets and music features, no significant difference is observed when kNN300 is paired with a numerical feature.

The results so far show that combining different scores can help to increase the accuracy. In the next set of measurements, we assess if the combination of scores also impacts other possible quality factors like diversity and coherence of the playlist continuations, which are in the focus of the subsequent optimization process.

4.1.2 Diversity and Coherence - Tags and Artists

Method. Our first method to assess the diversity and coherence is based on the artists and the social tags of the tracks. We use the *inverse ILS* to quantify the diversity level and – as a proxy – the *overlap* of artists and tags in the history and the playlist continuation to assess the coherence level.

Results (Tags). Table 4 shows the results[9] for the diversity and coherence in terms of social tags. We can observe that the similarity-based methods (Content, ContentPers) by design lead to the lowest diversity (highest homogeneity). The most diverse (inhomogeneous) playlists were generated by the popularity-based method and the nearest-neighbor approach kNN300. Combining the kNN method with content information and the single-artist heuristic in kNN300CPA – the most accurate method – leads to values between these extremes and comparably high homogeneity[10].

kNN generally leads to good coherence results in terms of the tags. Adding the single-artist heuristics can help to further increase the coherence significantly for Last.fm and AotM. This indicates that tracks by the same artist receive the same tags (e.g., genres, album name), which is supported by the fact that CAGH also leads to high coherence. The content-based recommendations are homogeneous in themselves, but the overlap with the playlist beginning is comparably low. The most accurate kNN300CPA method yields comparable or slightly less coherent lists than kNN alone.

Results (Artists). Table 5 shows the artist diversity and coherence. The "greatest hits" methods (SAGH, CAGH) by design lead to the lowest diversity and PopRank to the high-

Table 3: Recommendation accuracy results.

Playlist Algorithm	Accuracy @100 (hit rate | MRR)					
	Last.fm		AotM		8tracks	
PopRank	.029	.002	.048	.004	.050	.005
SAGH	.317	.100	.072	.014	.059	.008
CAGH	.334	.059	.081	.009	.061	.007
kNN300	.324	.067	.095	.010	.087	.007
Content	.236	.059	.078	.011	.048	.005
ContentPers	.236	.058	.079	.010	.048	.005
kNN300RY	.324	.064	.097	.011	.087	.007
kNN300TMP	.325	.067	.099	.010	.085	.007
kNN300LD	.323	.066	.099	.010	.086	.006
kNN300ART	.349	.072	.118	.011	.088	.007
kNN300CP	.347	.100	.116	.020	**.102**	**.009**
kNN300CPA	**.359**	**.102**	**.130**	**.022**	**.102**	**.009**
kNN300ALL	.356	.088	.111	.020	.099	**.009**

Table 4: Tag diversity and coherence.

Playlist Algorithm	Tag (inv. ILS | overlap)					
	Last.fm		AotM		8tracks	
PopRank	**.700**	.231	.548	.338	.651	.286
SAGH	.496	.357	.586	.405	.621	.385
CAGH	.482	.498	.579	.458	.621	.433
kNN300	.572	.477	**.706**	.475	**.726**	.523
Content	.116	.386	.153	.401	.144	.361
ContentPers	.117	.385	.149	.400	.143	.362
kNN300ART	.547	**.502**	.701	**.481**	.726	**.523**
kNN300CP	.321	.476	.334	.446	.479	.484
kNN300CPA	.339	.497	.343	.453	.479	.485

[7] Setups with more than one hidden track are possible. Another possible measure would be the "Average Log-Likelihood" [14]. We will not use it here because of the limitations discussed in [4].

[8] 8tracks allows two songs from any artist per playlist. Thus the artist heuristic has no effect on the accuracy of kNN300CPA vs. kNN300CP.

[9] We only include the most relevant algorithms here and not hybrids which are optimized toward other goals like, e.g., kNN300RY.

[10] This is similar to the tendency of the commercial playlister of "The Echo Nest", which generates lists with lower diversity than kNN [10].

Table 5: Artist diversity and coherence.

Playlist Algorithm	Artist (inv. ILS | overlap)					
	Last.fm		AotM		8tracks	
PopRank	**.817**	.009	**.907**	.031	**.994**	.019
SAGH	.251	**.628**	.597	**.416**	.694	.402
CAGH	.319	.590	.623	.393	.711	**.413**
kNN300	.496	.427	.788	.230	.983	.274
Content	.614	.258	.763	.109	.780	.078
ContentPers	.619	.251	.758	.108	.778	.076
kNN300ART	.418	.508	.771	.244	.981	.276
kNN300CP	.453	.433	.759	.192	.961	.227
kNN300CPA	.385	.510	.741	.213	.960	.229

est diversity. All kNN-based methods exhibit a tendency to place a mix of different artists in their lists[11]. The content-based method leads to varying results. Sometimes, the artist diversity is higher than kNN (Last.fm), sometimes it is equal or lower (AotM, 8tracks). This phenomenon depends on the absolute diversity level achieved by the kNN method, which is comparably low for Last.fm, where users often only include popular tracks of a few artists in their playlists. The hybrid methods kNN300CP and kNN300CPA often lead to an even lower diversity than their components alone, which means that the individual components select tracks from an overlapping set of artists.

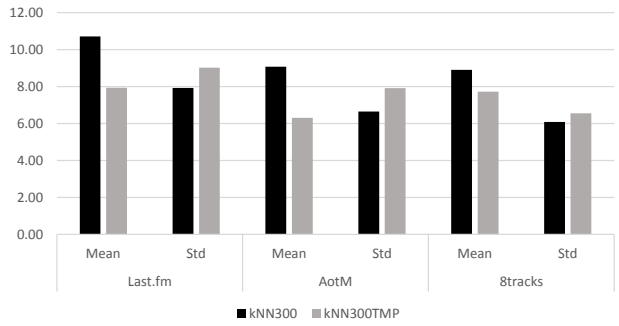
Discussion. Due to its hybrid nature the highly-accurate kNN300CPA method creates more homogeneous playlists than the pure kNN method in terms of tags and artists. At the same time, the playlist continuations generated by kNN300CPA are often less coherent than when using kNN alone, except for the Last.fm dataset. One possible explanation can be that the social tags used for tracks in the AotM and 8tracks datasets are more often related to the genre or mood and not to the artists. The content-based component would then push tracks that have the same genre but not necessarily the same artists as the history. In addition, the fact that the social tags for all datasets were obtained via the Last.fm API leads to a more sparse data situation for the other datasets.

4.1.3 Diversity and Coherence - Numerical Features

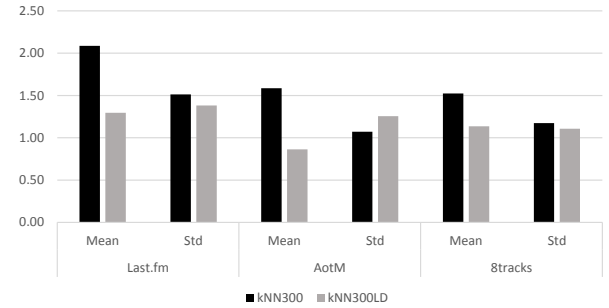
Method. For numerical features like the tempo, we analyze the effects of combining the kNN method with the respective scoring component, e.g., kNN300TMP. Specifically, we measure the *difference* between the mean value of the given feature in the history h and the generated playlist. To be successful, a method like kNN300TMP should result in smaller differences than when using kNN alone, i.e., the resulting playlist should match the tempo of the history better. Besides the mean value, we also report the change of the differences in the standard deviations to see if the distribution of the feature matches the history.

Results. Figures 1a to 1c show the results for the features tempo, loudness and release years. Across all dimensions and on all datasets, including a corresponding scoring component (significantly) helps to make the playlist continuation – in terms of mean – more coherent with the recent history than when using only the kNN scorer. At the same time, as shown in Table 3, adding the specific scoring method can furthermore often help to slightly increase the accuracy.

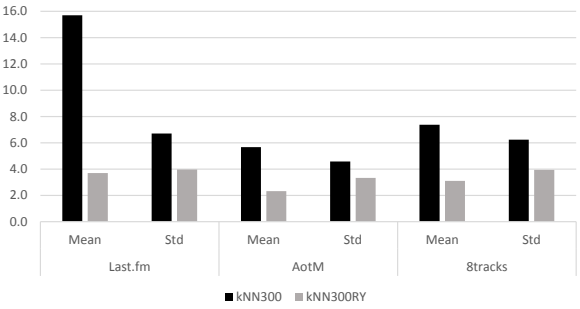
[11] A similar tendency is exhibited by "The Echo Nest" playlister [10].



(a) Difference of mean and deviation: kNN300 vs. kNN300TMP



(b) Difference of mean and deviation: kNN300 vs. kNN300LD



(c) Difference of mean and deviation: kNN300 vs. kNN300RY

Figure 1: Effects of combining the kNN-score with the scoring components for numerical feature.

The observations for the standard deviations (Std) are varying. The "error" for the release years and the loudness decreases – which means that we better match the feature's distribution in the history – or roughly remains the same. However, with respect to the tempo, we see that while the tempo scorer can better adapt to the average tempo, it varies more strongly than when we simply use the kNN method.

4.2 Results – Next-Track Optimization

Method. We applied the next-track optimization to the recommendations obtained with kNN300CPA, i.e., the approach with the best accuracy. Our goal is to analyze (a) if our method is effective in adapting the top-10 next-track list to the recent history and (b) how the adaptation affects the accuracy compared to kNN300CPA without optimization. The first aspect is assessed by measuring if the resulting playlists are "closer" to the history in different quality dimensions before and after optimization. For the latter aspect, we measured the accuracy before and after optimizing the result of kNN300CPA using Algorithm 1, this time at list length 10 to see possible effects for the immediate next

tracks. We tested several individual and combined optimization goals and again applied four-fold cross-validation.

Results (AotM). The detailed results obtained for a set of selected configurations for the AotM are shown in Table 6 In the first column, we list the evaluation goal(s), e.g., artist (ART) or tag (TAG) diversity, or popularity (POP). In the following seven columns, we report the *relative improvement* of the deviation from the history with respect to the baseline method kNN300CPA. The value ".27" in the upper-left corner therefore means that after optimization for artists, the resulting playlist was 27% closer to the history (i.e., a better continuation) in terms of artist diversity than the playlist generated by the baseline method kNN300CPA. The last four columns refer to the *relative change* in accuracy and the absolute accuracy values after optimization. The value ".28" in the first row means that optimization not only helped to better match the artist diversity, but also led to a 28% improvement in the hit rate (HR).

In the first seven lines of Table 6, we report the results of single-goal optimization for the AotM dataset. All individual optimizations were (significantly) effective, which can easily be seen as all values on the diagonal are positive. The improvements range from 17% (tag diversity) to 35% (track popularity). Regarding the accuracy results we can observe for all single-goal optimizations except the tempo that the optimization did not negatively impact accuracy. In contrast, the optimization in fact led to *relative improvements* both in terms of the hit rate (between 3 and 28%) and the MRR (up to over 100%).

Optimizing for tempo in the playlist continuation is effective to reduce the deviation by 31%, but led to a modest relative degradation of the hit rate and the MRR (6% and 1%). Some of the single-objective optimizations furthermore have some side-effects, for instance, they all have a positive effect on the popularity aspect. In addition, the genre, tag, and artist optimization goals seem to be partially correlated.

In the last row of Table 6, we report the results of a multi-objective optimization run. We can observe that the results were improved in all dimensions – although to a smaller extent than with single-objective optimization – and again even helped to improve the accuracy. Other multi-objective combinations are possible, but are sometimes affected by optimization trade-offs and a loss in accuracy.

Observations - Other datasets. The results for the non-public 8tracks dataset are in line with AotM. All optimizations were effective and the relative improvements are in nearly all dimensions even stronger than for AotM. The hit rate improvements, for example, ranged between 7.8% and 39% and no negative effects on accuracy were observed.

On the Last.fm dataset, again all single-goal optimizations were effective. With respect to accuracy the effects of optimizing for one goal are in some but not all cases positive. In particular optimizing for the tempo and the loudness can lead to a degradation of the hit rate of up to 16%. The MRR value is also sometimes negatively affected. A trade-off between the goals can be observed for multiple-goal optimizations and therefore not all possible combinations lead to an increase in accuracy.

The conclusion that can be drawn from this phenomenon is that criteria other than tempo and loudness seem to be the driving factors for users of the Last.fm platform when they create their playlists. For the users of 8tracks, matching both quality factors seems important; for the AotM users, an appropriate loudness seems more desirable. The observed differences between Last.fm and the other platforms are somehow expected, as 8tracks and AotM are platforms specifically designed for sharing playlists among music enthusiasts who consider various factors during playlist creation as analyzed in [7]. In contrast, users of Last.fm often seem to create recent "personal favorite tracks" playlists to be used by themselves.

Table 6: Next-Track optimization results (AotM). Target (= optimization target) can be ART = artist, TAG = tag, GNR = genre, POP = popularity, RY = release year, TMP = tempo, LD = loudness, [C] = Combination of [ART GNR POP RY]. Algorithm Parameters: Stopping criterion: Error reduction rate < 0.001, $maxSteps = 500$, $n = 10$, $m = 20$

Target	Relative improvement w.r.t. baseline [kNN300CPA]									Absolute	
	diversity			average				accuracy@10		accuracy@10	
	ART	TAG	GNR	POP	RY	TMP	LD	HR	MRR	HR	MRR
ART	.27	-.02	.14	.10	-.03	-.03	-.08	.28	1.17	**.072**	**.041**
TAG	.16	.17	.19	.16	-.02	-.06	-.09	.03	.65	.058	.031
GNR	.28	.04	.33	.12	-.02	-.04	-.07	.25	.61	.070	.030
POP	.02	-.04	.01	.35	-.07	-.06	-.06	.06	.70	.060	.032
RY	-.01	-.03	-.01	.10	.26	-.06	-.09	.19	.48	.070	.028
TMP	-.05	-.03	-.02	.11	-.10	.31	-.03	-.06	-.01	.053	.019
LD	-.03	-.02	.00	.11	-.08	-.04	.33	.16	.19	.063	.022
[C]	.22	.01	.22	.30	.01	-.05	-.07	.06	.75	.055	.021

5. RELATED WORK

A number of playlist generation techniques have been proposed over the last decade. They base their recommendations, e.g., on track co-occurrences and collaborative filtering, feature similarities and case-based reasoning, Markov models and sequential patterns, and various (discrete) optimization techniques [4]. In our work, we combine an unpersonalized co-occurrence based technique with content information as in [8] but furthermore (a) add personalization and feature-based scores in a weighted approach and (b) apply a post-processing procedure to match the characteristics of the immediate next tracks with the recent listening history.

From the evaluation perspective, we adopt a multi-metric and trade-off based evaluation approach as advised or applied for general and music-specific recommendation scenarios, e.g., in [1, 9, 11, 18], or [21]. In contrast to previous works, we however do not assume a "globally" desired level of diversity, novelty, or serendipity but aim to match the playlist continuation with the characteristics of a given listening history while maintaining accuracy.

Regarding our re-ranking approach, the work presented in [1] is similar to ours in that it tries to re-rank the first n items of an accuracy-optimized list in a way to increase or balance other quality factors. Their work however focuses on one single factor, "aggregate diversity", and aims to push items from the long tail. Diversity optimization – this time based on the ILS – was also done in [5] and [22] using techniques that reorder the recommendations based on their dissimilarity to each other. In [20], a binary optimization problem is used to model the trade-off between accuracy and diversity. Our approach can be used to achieve similar goals. It is however more generic in the sense that it is not limited to diversity-based measures and can deal with multiple goals at a time. Furthermore, we aim to match the diversity level of a given set of tracks or an individual user.

A comparable individualized "per-user" optimization technique for numerical quality factors was introduced in [9]. The authors employ a multi-objective linear optimization

approach to re-rank the whole recommendation list by changing relevance scores of a baseline algorithms. Like our next-track optimization, the baseline algorithm is exchangeable, it however requires the existence of scores which might not be available if the underlying technique optimizes a rank measure as in learning-to-rank approaches. Furthermore, the focus of [9] lies on promoting long tail recommendations and does not cover aggregate measures. Finally, our re-ranking technique can be configured to exchange elements greedily from a comparably small exchange set x_m, thereby leading to a limited computational complexity.

In [21], music tracks are recommended by combining multiple algorithms through rank interpolation. Similar to our faceted scoring method, their goal is to increase the recommendation accuracy as well as other factors like novelty, diversity and serendipity. However, their main focus lies on creating more serendipitous recommendations and the technique is not based on exchangeable baseline methods but implements the optimizations inside the algorithms. In addition, their method aims to achieve globally defined quality levels whereas in our work we try to minimize the difference between the playlist history and the immediate next tracks.

6. CONCLUSIONS

Playlist generation is a special form of music recommendation which is particularly challenging as the recommended items are immediately "consumed", sequentiality and homogeneity aspects can be important, and the choice of a few wrong tracks within the sequence can easily be detrimental to the perceived service quality as a whole.

In our work, we have proposed a new multi-aspect scoring scheme which combines patterns in existing playlist with meta-data features and personal user preferences to achieve higher accuracy and, if desired, higher homogeneity than previous approaches. A post-processing procedure can furthermore help to match the characteristics of the playlist continuation with the most recent listening history. In our ongoing work we evaluate the application of these procedures to general recommendation settings, where, e.g., the most frequent user ratings are used as representatives for the recent user preferences to which the next item recommendations should be adapted. Conducting a user study to evaluate our approaches in terms of the perceived quality of the playlists and user satisfaction is part of our future work.

7. REFERENCES

[1] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE TKDE*, 24(5):896–911, 2012.

[2] W. Balkema and F. van der Heijden. Music Playlist Generation by Assimilating GMMs into SOMs. *Pattern Recognition Letters*, 31(11):1396–1402, 2010.

[3] G. Bonnin and D. Jannach. Evaluating the Quality of Playlists Based on Hand-Crafted Samples. In *Proc. ISMIR*, pages 263–268, 2013.

[4] G. Bonnin and D. Jannach. Automated generation of music playlists: Survey and experiments. *ACM Comput. Surv.*, 47(2):26:1–26:35, 2014.

[5] K. Bradley and B. Smyth. Improving Recommendation Diversity. In *Proc. AICS '01*, pages 75–84, 2001.

[6] P. Y. K. Chau, S. Y. Ho, K. K. W. Ho, and Y. Yao. Examining the effects of malfunctioning personalized services on online users' distrust and behaviors. *Decis. Support Syst.*, 56:180–191, 2013.

[7] S. Cunningham, D. Bainbridge, and A. Falconer. 'More of an Art than a Science': Supporting the Creation of Playlists and Mixes. In *Proc. ISMIR*, pages 240–245, 2006.

[8] N. Hariri, B. Mobasher, and R. Burke. Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns. In *Proc. RecSys*, pages 131–138, 2012.

[9] T. Jambor and J. Wang. Optimizing multiple objectives in collaborative filtering. In *Proc. RecSys '10*, pages 55–62, 2010.

[10] D. Jannach, I. Kamehkhosh, and G. Bonnin. Analyzing the characteristics of shared playlists for music recommendation. In *RSWeb Workhop at ACM RecSys '14*, 2014.

[11] D. Jannach, L. Lerche, F. Gedikli, and G. Bonnin. What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects. In *Proc. UMAP 2013*, pages 25–37, 2013.

[12] M. Kaminskas and F. Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(2–3):89–119, 2012.

[13] B. Logan. Content-Based Playlist Generation: Exploratory Experiments. In *Proc. ISMIR*, pages 295–296, 2002.

[14] B. McFee and G. R. Lanckriet. The Natural Language of Playlists. In *Proc. ISMIR*, pages 537–542, 2011.

[15] B. McFee and G. R. Lanckriet. Hypergraph Models of Playlist Dialects. In *Proc. ISMIR*, pages 343–348, 2012.

[16] S. Pauws, W. Verhaegh, and M. Vossen. Music playlist generation by adapted simulated annealing. *Inf. Sci.*, 178(3):647–662, 2008.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. UAI '09*, pages 452–461, 2009.

[18] A. Said, B. J. Jain, and S. Albayrak. A 3D approach to recommender system evaluation. In *Proc. CSCW '13 Companion Volume*, pages 263–266, 2013.

[19] A. M. Sarroff and M. Casey. Modeling and Predicting Song Adjacencies In Commercial Albums. In *Proc. SMC*, 2012.

[20] M. Zhang and N. Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *RecSys '08*, pages 123–130, 2008.

[21] Y. C. Zhang, D. O. Séaghdha, D. Quercia, and T. Jambor. Auralist: Introducing serendipity into music recommendation. In *Proc. WSDM '12*, pages 13–22, 2012.

[22] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proc. WWW '05*, pages 22–32, 2005.