EVALUATING THE QUALITY OF PLAYLISTS BASED ON HAND-CRAFTED SAMPLES

Geoffray Bonnin
TU Dortmund, Germany
geoffray.bonnin@tu-dortmund.de

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

ABSTRACT

The automated generation of playlists represents a particular type of the music recommendation problem with two special characteristics. First, the tracks of the list are usually consumed immediately at recommendation time; second, tracks are listened to mostly in consecutive order so that the sequence of the recommended tracks can be relevant. A number of different approaches for playlist generation have been proposed in the literature. In this paper, we review the existing core approaches to playlist generation, discuss aspects of appropriate offline evaluation designs and report the results of a comparative evaluation based on different data sets. Based on the insights from these experiments, we propose a comparably simple and computationally tractable new baseline algorithm for future comparisons, which is based on track popularity and artist information and is competitive with more sophisticated techniques in our evaluation settings.

1. INTRODUCTION

Among the different application domains of recommender systems (RS), music is often considered as being particularly difficult to deal with [5, 12]. One specific approach for music recommendation and discovery is the automated generation and provision of playlists (mixes). This strategy however induces additional challenges as the tracks are consumed in sequence and usually immediately at recommendation time. This means that the context of the previous recommendations has some influence on user satisfaction and should be taken into account in the playlist generation process.

When our goal is to automatically generate playlists, i.e., lists of sequentially ordered tracks, one key question is the evaluation of their quality. In fact, there might be a number of different factors that influence the perceived value of a playlist, including, e.g., the coherence of the list, or the variety or *freshness* of the songs [9]. Many playlist generation algorithms have been proposed in the literature, but their quality is hard to compare as they often focus on particular families of techniques and use different baseline algorithms and evaluation criteria.

In this paper, we review existing playlist generation approaches and propose a comparably simple and computationally efficient new baseline algorithm for future comparisons which relies on track popularity and artist information. We discuss evaluation designs from the literature and present experiments that use two evaluations metrics based on comparisons with hand-crafted playlists, i.e., playlists made by hand by music enthusiasts. The results show that our algorithm outperforms the other approaches on two out of three data sets when using hit rates as a metric. In addition, the experiments reveal a considerable limitation of using the log-likelihood metric as a means to compare the quality of automatically generated playlists.

2. AUTOMATED PLAYLIST GENERATION

In the following, we review existing approaches to playlist generation and present our new approach. A playlist is usually defined to be an ordered sequence of musical tracks. The playlist generation problem typically consists in creating such a list given either some *seed* information or semantic description [3]. As another input, we might also have some extra information for each track, e.g., the audio signal, the composer, artists, lyrics, tags, ratings, etc.

In this work, we assume that the seed information consists of the listening *history* so far. Given this history, the system presents recommendation lists of tracks to the user, and each time a track is selected the process is repeated [4]. Thus, the problem comes down to the computation of the score of a candidate track t given a playlist history $h = \langle t_1, t_2, ..., t_i \rangle$. The resulting scores – which in some cases correspond to probability estimates – can then be used to filter and rank the remaining tracks.

2.1 Markov chains

Attempting to recommend tracks that represent a smooth transition from the previous track is an obvious approach. This corresponds to the Markov property and leads to a first-order Markov model in which states correspond to tracks. Given a history h of a playlist and a candidate track t, the probability of t in such a model thus only depends on t_i, the last element of h. Examples of playlist modeling approaches based on this strategy include [14] and [6]. In [14], the authors compare a set of approaches to assign transition probabilities to a Markov model including the uniform distribution, tags, audio signal and artist information. In [6], a more sophisticated **Latent Markov Embedding** (**LME**) model in which tracks are represented by

vectors in the Euclidean space is compared with the bigram model and a uniform distribution.

The major limitation of these models is that the assumptions on which they are based may be too strong as the choice of the next track by a user may or may not depend only on the previous track. Although tracks are usually being listened one after the other and transitions between tracks surely have some importance, in practice, the rules users follow to build playlists can be quite different and often contradict this assumption, see also [8].

2.2 Frequent patterns

Another possibility to recommend tracks for playlist generation is to extract frequent patterns from playlists. The common techniques are **association rule** (**AR**) and **sequential pattern** (**SP**) mining. An association rule [1] has the form $A \rightarrow C$, where A and C are two itemsets. *Sequential patterns* are a sequential version of association rules [2] in which the order of the elements in the pattern is also taken into account in the mining process. The additional constraints of sequential patterns over association rules can in general lead to more accurate recommendations, but the approach has a higher computational complexity and requires a larger amount of training data. Another possible limitation of this approach might be the comparably small confidence values for the extracted patterns given the usually high sparsity of musical data sets.

2.3 Neighborhood recommenders

Another way to exploit co-occurrences of tracks is to use a ***k*-nearest-neighbors** (***k*NN**) recommender which is based on the similarity between playlists. Such a kNN approach was proposed in [11] and used as a basis for a more sophisticated recommender which uses sequential patterns of latent topics based on tags. Similar to association rules, this kNN approach not only exploits information about the collocation of items in playlists but also takes the number of shared items in each playlist into account when estimating the probability. However, association rule mining is based on counting the frequency of patterns for all users in an offline process. The kNN approach, in contrast, dynamically computes a "local" probability using the k most similar playlists. In other words, the limitation mentioned in the previous subsection with respect to low confidence values is reduced. The computation of neighborhoods and playlist similarities is however computationally complex both in terms of time and space, making the approach intractable when recommendations have to be made in real time.

2.4 Playlists as users

In principle, if we interpret the playlist generation problem to be similar to the item prediction problem in typical RS settings by considering playlists to be users, existing RS algorithms for item recommendations can be applied including recent learning-to-rank techniques. In particular, the BPR-approach (Bayesian Personalized Ranking) from [15] has been included in previous comparative evaluations for playlist or music recommendation, see e.g. [11] and [13]. The experiments in the last two papers however show that the plain BPR method can be easily outperformed by other methods in particular problem settings.

2.5 Content-based approaches

Using additional information, one can try to enhance the confidence of pattern-based approaches or avoid the complexity of the kNN approach. Such additional information can be the content of the tracks (lyrics, spectrum, etc.), the similarity of musical features [10], user tags, or more simple elements such as artist names. Some of the aforementioned approaches use some forms of content and metadata. For instance, the topic-aware hybrid recommender of [11] uses tags to determine topics, but does not solve the scalability problem of the underlying kNN approach. Also McFee and Lanckriet [14] experiment with Markov models that use tags, the audio signal and artist names. However, their approach does not solve the problem of the strong assumption of the Markov property.

Regarding the incorporation of additional information into the recommendation process, we hypothesize that the use of artist names in general is particularly promising as this type of data is objective, easy to obtain and to process (as opposed to, for instance, information about the playlist topic, genre or style).

2.6 Popularity-based approaches

In many application domains for RS and in particular in the music domain [5], we can observe a so called "long tail" distribution of items, meaning that a small subset of the items accounts for the majority of transactions or interactions. This popularity bias results in the fact that simple popularity-based approaches, which present the same set of popular items to everyone, can represent a comparably hard baseline [7]. Given these observations, we included two approaches that are based on popularity combined with artist information in the experiments.

"Same artists - greatest hits" (SAGH): In [13], the authors propose a baseline algorithm for music recommendation – not in the context of playlists – called "Same artists - greatest hits", which simply recommends the most popular songs of the artists appearing in the user's listening history. Their experiments on the Million Song data set shows that higher prediction accuracy can be obtained with such an approach than when using, e.g., the above-mentioned BPR method. This method would thus be a hybrid that uses both additional information as well as popularity.

"Collocated artists - greatest hits" (CAGH): In this paper, we do not only apply the previous scheme, but propose an extension to it. Our assumption is that the different artists that are included in playlists by the users are not too different from each other. We thus propose to recommend tracks based on the frequency of the collocation of artists.

More precisely, we compute the similarity between two artists a and b according to the following formula:

$$\text{sim}_{\text{a}}(a,b) = \frac{\sum_p (\delta_{a,p} \cdot \delta_{b,p})}{\sqrt{\sum_p \delta_{a,p} \cdot \sum_p \delta_{b,p}}}$$

with $\delta_{a,p} = 1$ if playlist p contains a and 0 otherwise. The similarity thus depends on the collocations of artists within playlists, which can be computed offline. Our proposed formula for the computation of the score of a next track t with artist a given a playlist beginning h is as follows:

$$\text{score}_{\text{CAGH}}(t,h) = \sum_{b \in A_h} \text{sim}_{\text{a}}(a,b) \cdot \text{counts}(t) \quad (1)$$

where A_h is the set of artist names of the tracks in h and $\text{counts}(t)$ is the number of occurrences of t in the data set, which corresponds to the greatest hits of the data set.

3. A COMPARATIVE EVALUATION OF PLAYLIST GENERATION STRATEGIES

The presented playlist generation techniques follow different strategies and exploit different types of inherent characteristics of the playlists or rely on external information. The goal of this study is to obtain a better understanding of different aspects related to the generation and evaluation of playlists. In particular, we aim to better understand the role of sequentiality and popularity, find out if different metrics follow the same trends in a comparative evaluation and if the observations are consistent across different data sets.

3.1 Data Sets

We used three data sets in our experiments. One is from Artofthemix, which is probably the most commonly used data set for related research [11, 14]. The second was retrieved from last.fm [1]. The third was provided to us by 8tracks [2]. In order to reduce the sparsity of the data, we used the web service of Musicbrainz [3] to correct artist and track misspellings. We also removed playlists of size 1. For the last.fm data, we furthermore decided to select playlists in a way that long-tail tracks are used at least twice. Table 1 shows the data set characteristics.

Notice that the Artofthemix data does not contain user IDs. As implicitly done also by [11], we consider users as being equivalent to playlists, as they usually do not create large numbers of playlists. Regarding track occurrences, the last.fm and 8tracks data sets have a similar average track usage count (5.5 and 5.3) [4]. This usage count is significantly smaller for Artofthemix (2.7). Another related characteristic is the long tail distribution of track usages. Table 1 divides the corresponding distribution into three parts: head, middle and tail. The "head" contains tracks which appeared more than 20 times in playlists, tracks in the "middle" were included in playlist between 2 and 20 times, and songs from the "tail" were only used once or twice. These values are admittedly somewhat arbitrary but allow us to roughly compare the respective distributions. The resulting proportions reveal another difference between the last.fm and 8tracks data sets: although they have a similar average track usage count, the size of the long tail of 8tracks is much larger.

[1] http://www.lastfm.com/api
[2] http://8track.com
[3] http://musicbrainz.org/ws/2/
[4] The track/artist usage count means how often a track/artist was used in all playlists

	last.fm	Aotm	8tracks
Playlists	50,000	28,636	99,542
Users	47,603	–	51,743
Tracks	69,022	214,769	179,779
Avg. tracks/playlist	7.6	20.1	9.7
Avg. track usage count	5.5	2.7	5.3
Head	4.8%	1.6%	4.5%
Middle	35.0%	18.5%	25.7%
Tail	60.1%	79.9%	69.8%
Artists	11,788	47,473	29,352
Avg. artists/playlist	4.5	17.3	8.9
Avg. artist usage count	32.2	12.1	32.7
Artist reuse rate	31.1%	21.8%	13.8%

Table 1. Properties of the data sets.

Regarding artist-based recommendation approaches, Table 1 shows that playlists usually contain fewer artists than tracks. The row "artist reuse rate" in the table represents the percentage of cases when the artist of the last track of a playlist already appeared in the same playlist before. The corresponding values are 31.1% for the last.fm data set, 21.8% for the Artofthemix data set and 13.8% for the 8tracks data set. This represents another difference between the last.fm and 8tracks data sets: although they have a similar average artist usage count, the artists are more distributed across the playlists in the 8tracks data because 8tracks' license only allows for up to two songs from the same artist per playlist. Overall, we think that these values represent a strong argument to emphasize on artist names as an additional information when recommending tracks, except maybe for the 8tracks data set.

Using three data sets with quite different characteristics should allow us to analyze how the different algorithms perform in different situations. In general, generating recommendations based on the Artofthemix data set should be much more difficult than with the last.fm and 8tracks data sets, as it is smaller and the individual tracks are less often used. Other factors may however play a major role as well, in particular the size of the long tail.

3.2 Evaluation Metrics

Playlist generation is usually evaluated according to three possible strategies: semantic cohesion, human opinion survey and comparison with hand-crafted playlists. Semantic cohesion corresponds to the assumption that a good playlist is a playlist which tracks are as homogeneous as possible, e.g., in terms of genre or style, which can be considered as a strong assumption. Human opinion surveys do not have this drawback but are time consuming and difficult to reproduce. We thus choose the third option and

evaluate playlisters by comparing their output with handcrafted playlists. Two evaluation strategies are common, the hit rate and the average log-likelihood.

3.2.1 Measuring hit rates

A first way of measuring the accuracy of playlist generation for music recommendation is to use the *hit rate*, i.e., the proportion of relevant predictions on a test set. An evaluation method of this type is used, e.g., by [11], who hide the last element of each given playlist, which has then to be recommended by the algorithm. In general, any subset of playlist elements could be hidden in such a protocol. Removing the last one however is based on the assumption that the sequential history of a playlist can be relevant.

The limitation of this evaluation metric is that it corresponds to the assumption that the actual next tracks in the playlist are the only relevant tracks that can be recommended, although some other tracks may be relevant. In other words, it is possible that hundreds of tracks are relevant, but as the recommender has to select a subset of them, the actual next tracks of the test playlists might not be recommended. As it is impossible to know how many tracks are relevant for each situation, it is reasonable to analyze the accuracy of a system using longer recommendation lists. Such lists could of course not be used in a real framework, but our goal here is only to compare the algorithms. The assumption is then that there is a correspondence between the size of the recommendation lists and the average number of relevant tracks. Still, the hit rate can only be considered to be a lower bound for the accuracy.

3.2.2 Measuring the average log-likelihood

Another way to measure accuracy is to use the *average log-likelihood*. The average log-likelihood can be used to measure how likely a system is to recommend the tracks of a given set of playlists through a weighted random process. More precisely, given a test set of playlists, the average log-likelihood can be determined by computing the probability of observing each next track according to the corresponding playlist history and some model learned on the training data. Research on music recommendation using playlists that use this metric includes [6] and [14]. Obviously, the application of this measure requires that the output of a playlist recommender can be expressed as probability values for each song, which can be easily obtained by a normalization over the prediction lists.

In contrast to the hit rate, which provides a realistic lower bound on the accuracy that is directly interpretable, this metric is not interpretable on an absolute scale: the possible values vary between $-\infty$ (at least one track in the test set has a corresponding 0 probability in the model) and 0 (all probabilities in the model for all tracks in the test set are 1). Thus, this metric does not tell us if a generative approach leads to good playlists, but allows us to compare the results of different generative approaches. It can thus be considered as a complementary measure to the hit rate.

As only one track having a 0 probability is sufficient to induce a $-\infty$ average log-likelihood, 0 probabilities must be avoided. This requires an additional smoothing step, which might result in a strong bias. For instance, new tracks will always have such 0 probabilities with a frequency-based approach. In that situation a combination with the uniform distribution can be used, but then the weight of the uniform distribution may become too large given the long-tailed distribution of our data sets.

3.2.3 Computational complexity

Another important fact that should be taken into account is the computational complexity. Indeed, as opposed to, for instance, movie recommendation, for which recommendations can be computed offline and updated regularly, music recommendation can be highly dynamic and contextual. Users usually listen to tracks in sequence, where each track lasts a few minutes. Therefore, a music recommender should be able to provide fast contextual recommendations. Moreover, as the number of tracks that can be recommended is usually very high, the efficiency of the training phase can become crucial. In the subsequent analysis of algorithms, we will thus also briefly discuss aspects of computational complexity.

4. EXPERIMENTS

In the following evaluation, we use the two aforementioned accuracy metrics: hit rate and average log-likelihood. A 10-fold cross-validation procedure was applied for both metrics on the three data sets. Recall that the total number of tracks of the data set highly influences the hit rate values. In [11], the results for prediction lists of size varying between 1 and 300 given $21,783$ tracks are reported. This corresponds to the selection of about 1.5% of the tracks. We used a similar proportion in our experiments and set the maximum size of the prediction lists to $1,000$ for last.fm, $3,000$ for Artofthemix and $2,500$ for 8tracks.

4.1 Evaluating Hit Rates

Figure 1 shows the results of comparing five different recommendation approaches on the three data sets using the hit rate. The approaches include the three above-mentioned frequent-pattern approaches AR and SP, a kNN recommender using 50 and 100 neighbors, the SAGH recommender and our new baseline recommender CAGH [5].

We can first notice that all approaches lead to comparably low accuracy values for short recommendation lists. For longer recommendation lists, our new CAGH recommender clearly outperforms the other approaches on the last.fm and Artofthemix data, except for recommendation lists longer than $2,800$ for the frequent-patterns approach on the Artofthemix data. On the data from 8tracks, the frequent pattern approach clearly outperforms all other approaches, followed by the kNN approach with 100 neighbors, and the CAGH recommender. The reason for the lower performance of the CAGH recommender is probably the better distribution of artists across playlists on this particular data set due to the corresponding license restrictions (see section 3.1).

[5] The method of [11] is not included here but is comparable to the kNN method according to their measurements.

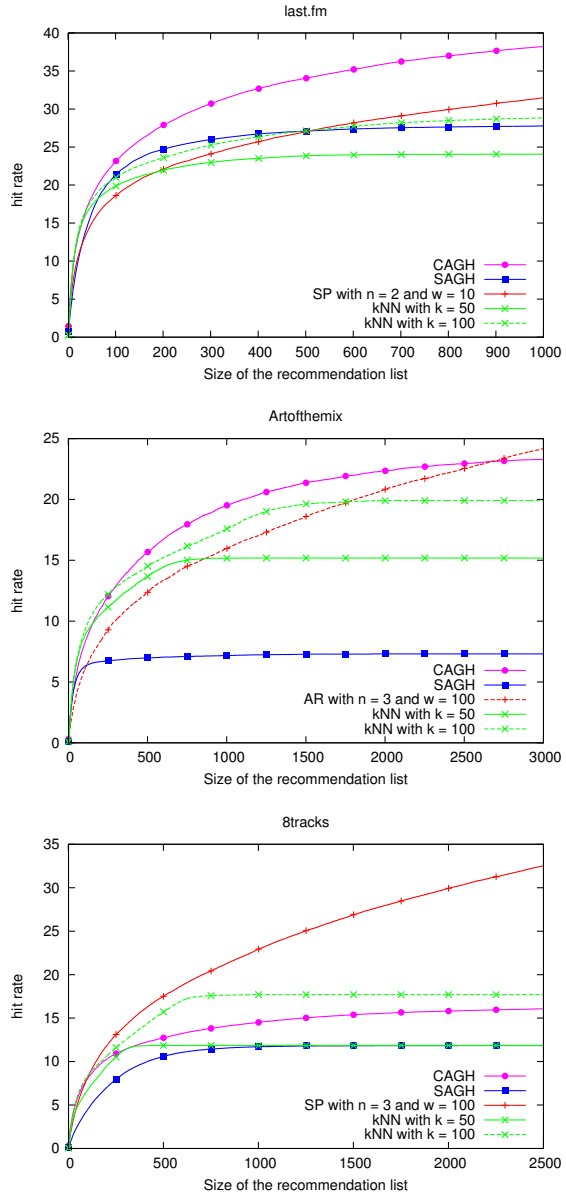

Figure 1. Hit rates of the different approaches.

In general, using more neighbors enhances the accuracy of the kNN approach on the three data sets. The kNN approach may even outperform all the other approaches using more than 100 neighbors. However, both neighborhood sizes used in these experiments are already high and make the recommendation algorithm not only intractable in terms of space requirements, but also in terms of running time. Still, kNN approaches lead to a lower accuracy values than both our new baseline approach and the frequent patterns method. More precisely, on the three data sets the accuracy of the kNN approach seems to be limited by the size of the recommendation lists it is able to build. This is probably the reason why the frequent patterns outperform this approach on the 8tracks data set, as it is close to a kNN approach that uses all the neighbors.

Other observations depend on the used data set. In particular, for the last.fm data set, the SAGH recommender leads to results that are similar to those of the kNN recommender with 100 neighbors. For the Artofthemix data set, the SAGH recommender is clearly outperformed by all other approaches. For the 8tracks data set, it leads to results that are similar to those of the kNN recommender with 50 neighbors for recommendation lists longer than 750.

Beside the results shown in Figure 1, we also experimented with models based on the Markov property, among them the simple bigram model and the recent Latent Markov Embedding (LME) model of [6]. Despite the long time that can be required to train these models – e.g., several weeks for the LME model – these methods led to particularly low accuracy values which were consistently below 10% for recommendation lists of size $1,000$ for the last.fm data set and 5% for recommendation lists of size $3,000$ for the Artofthemix data set. We therefore omit these results in this paper. In general, given these comparably strong differences, assuming the Markov property might be too strong for this problem setting. Furthermore, our results indicate that emphasizing on artist names can be particularly promising for accurate track recommendation in the context of playlist generation.

4.2 Evaluating Average Log-Likelihoods

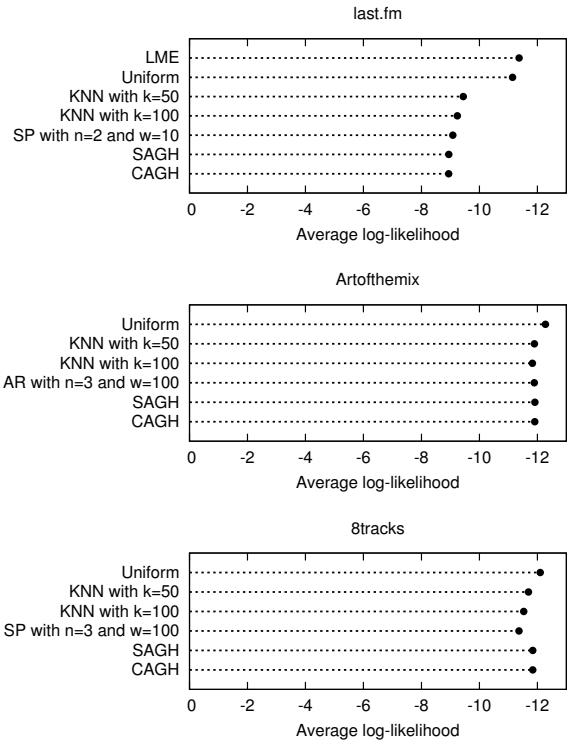

Figure 2. Av. log-likelihood of the different approaches

Figure 2 shows the results of the generative versions of the five approaches on the three data sets using the average log-likelihood. The experimented methods all correspond to mixture models that combine the uniform distribution with the approaches of the previous set of experiments. In order to perform these evaluations, each training set of the cross-validation process was further split into a learning set for obtaining model probabilities and a validation set to compute optimized weights using the Expectation-

Maximization algorithm. We also provide the results of the LME model with standard parameters on the last.fm data (on both other data sets the training process lasted more than a month).

We focus on the correspondences with the previous set of hit rate results. On the last.fm data set the mixture of the uniform distribution with the CAGH and SAGH recommenders leads to the best average log-likelihood values. It is worth noting that both models provide similar results according to this metric although on the same data set the CAGH recommender was a clear winner in terms of the hit rate. We also evaluated all the mixture models of this section in terms of the hit rate and obtained a similar output: both CAGH and SAGH recommenders lead to a similar hit rate when combined with the uniform distribution. More precisely, this combination strongly lowers the hit rate of the CAGH recommender, but not that of the SAGH recommender. This result confirms the bias of using this metric we mentioned in section 3.2: as a relatively important number of tracks are new in the test sets, EM-based mixtures tend to give more weight to tracks coming from the uniform distribution than tracks coming from the similar artists. This phenomenon also appears on the three data sets, although the CAGH and SAGH do not outperform the other models. On the Artofthemix data set, it even applies to all the models: the accuracy of all approaches is strongly lowered when combined with the uniform distribution. These results indicate a limitation of using the average log-likelihood metric, i.e., smoothing the models have highly lowered the accuracy, although in reality a RS does not have to avoid 0 probabilities.

When testing the LME model on the last.fm data set, the experiments showed that the model lead to lower results than the uniform distribution. This confirms the previous conclusion about the use of the Markov property.

5. CONCLUSION

This paper proposes a classification of existing approaches for playlist generation and discusses limitations of typical experimental designs, which for example do not take scalability aspects into account or are based on comparably strong assumptions such as the Markov property. Based on this discussion, we propose a new computationally efficient recommendation scheme based on popularity and artist information. An experimental comparative evaluation showed that our algorithm outperforms the other approaches in terms of hit rate on two of three data sets. On the remaining data set, our recommender is on a par with neighborhood-based approaches and was outperformed by a frequent pattern technique. This difference is probably caused by the high dispersion of artists among playlists due to license constraints. However, other factors may have induced this difference in accuracy, which we are investigating in our current work. Our evaluations also put forward a strong limitation of using the average log- likelihood metric: it implies to smooth models in order to avoid 0 probabilities which resulted in a strong degradation of the quality of the approaches.

6. ACKNOWLEDGMENTS

We thank 8tracks for providing us their valuable data.

7. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD 1993*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. ICDE 1995*, pages 3–14, 1995.

[3] L. Barrington, R. Oda, and G. Lanckriet. Smarter than Genius? Human Evaluation of Music Recommender Systems. In *Proc. ISMIR 2009*, pages 357–362, 2009.

[4] Dominikus Baur, Sebastian Boring, and Andreas Butz. Rush: Repeated Recommendations on Mobile Devices. In *Proc. IUI 2010*, pages 91–100, 2010.

[5] Ò. Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[6] S. Chen, J.L. Moore, D. Turnbull, and T. Joachims. Playlist Prediction via Metric Embedding. In *Proc. KDD 2012*, pages 714–722, 2012.

[7] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys 2010*, pages 39–46, 2010.

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

[9] B. Fields. *"Contextualize Your Listening: The Playlist as Recommendation Engine"*. PhD thesis, Goldsmiths, University of London, London, UK, April 2011.

[10] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist Generation Using Start and End Songs. In *ISMIR 2008*, pages 173–178, 2008.

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

[12] P. Lamere and Ò. Celma. Music Recommendation and Discovery Remastered, Tutorial at ACM RecSys 2011. Online at `http://www.slideshare.net/slideshow/embed_code/9860137`, 2011.

[13] B. McFee, T. Bertin-Mahieux, D. Ellis, and G. Lanckriet. The million song data set challenge. In *Proc. AdMIRe'12*, 2012.

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

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. UAI*, pages 452–461, 2009.