Mining Music Playlogs for Next Song Recommendations

Andre Busche, Artus Krohn-Grimberghe, Lars Schmidt-Thieme
University of Hildesheim, Germany
{busche, artus, schmidt-thieme}@ismll.uni-hildesheim.de

Abstract

Recommender systems are popular social web tools, as they address the information overload problem and provide personalization of results [1]. This paper presents a large-scale collaborative approach to the crucial part in the playlist recommendation process: next song recommendation. We show that a simple markov-chain based algorithm improves performance compared to baseline models when neither content, nor user metadata is available. The lack of content-based features makes this task particularly hard.

1 Introduction

Nowadays, mobile devices enable ubiquitous access to any piece of music any time and anywhere. Additionally, online stores offer millions of song tracks to those that want fresh content. But which of this plethora shall the store suggest or the user choose? How to cope with the information overload problem? Recommender systems come as a rescue: they aim at providing the user with relevant music content, based on personal listening preferences.

Given the usage pattern for mobile music devices, in the extremes two different scenarios emerge that music recommender systems should tackle: on the one hand, there is the interactive listening scenario, where users actively use the system and want to influence the next song being played; on the other hand, there is the passive listening scenario, where music is just an ambient factor and the users want to receive full playlists instead of single tracks.

In the interactive listening scenario, e.g., music online stores aim at increasing their revenue by recommending tracks to users who did not purchase these songs before. Another use case may be a music player automatically suggesting candidates for the next song to play.

In the passive listening scenario, portable devices are often used to listen to one's own music in a passive way while doing something else, e.g., sports. Collaborative listening portals like last.fm[1] or Pandora[2] automatically generate music playlists given some seed song or artist, and provide the user with a sequential but uncontrollable list of tracks to listen to. One of their aims is to get users in contact to novel tracks they did not know before, possibly with the aim to sell those.

'Automatic playlist generation' is involved in all of the cases above as a means of generating a probable sequence or set of songs to play (next) given some fixed but potentially huge set of songs in a library. While all these applications are of the same kind, the available data for them used to differ considerably: Online stores and listening portals usually had no information about the music library of the user and, traditionally, portable devices were only able to choose next tracks from the users' music library for playlist generation. The availability of nearly ubiquitous internet, though, changed this picture where services like iTunes or Zune in combination with the respective devices have access to both, the user's music library and the vast libraries of online stores.

In this paper, we exploit massive collaborative usage data gathered on portable devices to recommend songs a user might want to listen to next. In the recommendation lists, songs available in the user's library as well as songs not yet existing in his library are considered. Thus we allow for utility maximization of the user's library and for cross-selling opportunities for the music store the user is connected to.

We can show that collaborative recommendation techniques can greatly improve performance over baseline methods in situations where content features are unavailable. Furthermore, we demonstrate that for next song recommendation denser data yields better performance.

2 Related Work

Playlist generation has been studied with great interest for about the last five years by many researchers, each focusing on different recommendation techniques. A content-based approach to derive playlists by providing the system with one start and end song has been introduced in [3]. The authors strive at providing smooth transitions between genres: Having MFCC coefficients[4] from the start and end song at hand, they model the songs by a single Gaussian and compute an 'optimal transition playlist' by means of a (weighted) sum of 'optimal' divergence ratios. Figuratively, their approach recommends those songs which are closest to the λ-weighted 'connection line' between the songs. The evaluation in [3] is based on whether the generated playlists only contain songs from source genre, target genre, or both. However, it has been shown in [9] that while experts and social communities commonly agree when (manually) classifying songs between distinct genres at a coarse level (e.g. 'Classic', or 'Rock' songs), their agreement decreases when it comes to the classification of songs to subgenres.

An algorithm for learning possible song transitions from radio station playlists was introduced in [5]. While their approach is close to ours on a conceptual level, it is radically

[1] http://www.last.fm/

[2] http://www.pandora.com/

different w.r.t. the features used. They use content features such as the MFCC coefficients and others to model song transition probabilities based on the content, rather than collaborative play information and available metadata as in our approach. To this end, they are more concerned enabling their approach to also work with previously unknown content, for which no collaborative data yet exists. This problem, known as the 'new-item'-problem for recommender systems, as well as the 'new-user' problem [8], is not handeled here.

Additionally, [5] incorporate tag information when recommending playlist items. Tags have become a cornerstone of the Web 2.0 and, furthermore, they increase recommender performance for item prediction (e.g. next song prediction) [6]. A manual way to get 'gold-standard' tags is described in [10]. Regrettably, tags are not available in our dataset.

While there is much related work in the area of movie recommendation (e.g. [11]), we deem this task to be fundamentally different: One the one hand, short video clips and music tend to be consumed in a session-oriented way. On the other hand, TV programme length movies are mostly consumed one at a time.For the former scenario, it can be assumed that within a session a user's mood remains in more or less the same state. For the latter, the time spans are considered too large for this to be necessarily the case.

3 Methodology

In this section we describe our technique for next song recommendation based on playlogs. For training data we restrict our algorithms to collaborative data. We use the Zune dataset which was sponsored by Microsoft Zune[3]. It contains an anonymized sample of multimedia play log entries from Sep '08 to Feb '09 plus additional metadata on the multimedia items. With 1.3 billion log entries this dataset is quite large, leveraging information from 1.3 million users $u \in \mathcal{U}$ on 44 million song tracks $t \in \mathcal{T}$. Further does it contain playlists generated by external experts. These playlists comprise songs grouped by, e.g., themes. However, neither content-information nor tags are available in our sample.

For the remainder of this work, we only consider a subset of the data: First, we only use songs, ignoring the other multimedia content. Second, of the songs we restrict ourselves to those contained in the predefined playlists. Furthermore, we only consider users who have played at least 60 songs, possibly spread over multiple sessions $\mathcal{S}$. This filtering reduces the data to a 'dense' part, composed of approx. 500k users u and 26k tracks t. 18.4 million (1.4% of the original) log entries relate these users to items. This yields our largest training and sole evaluation dataset, 'Playlog'.

Our data also contains some hand-made playlists created by experts. Since the playlist information is explicitly given, we can directly transform them into sessions for later use. From the playlists, we derive two further datasets:

- 'Playlist (sequential)': The sequential occurrence within the playlist, as initially ordered by the expert.
- 'Playlist (combinatorial)': The set-based interpretation of the playlists.

We define the density of our data to be the fraction of present transitions to the amount of possible transitions

$$\text{density}(\mathcal{S}) := \frac{|\,\mathcal{S}\,|}{|\,\mathcal{T}\,|^2}$$

with $\mathcal{T}$ being the filtered set of tracks ($|\ \mathcal{T}\ | \sim 26k$). Let a 'transition' from song t_i to song t_{i+1} happen iff song t_i is played for its whole duration, and song t_{i+1} directly follows t_i, possibly with some lag (see below).

Table 1 gives an overview of the densities within our derived datasets.

Technique	Density
Playlog	2.2%
Playlists (sequential)	0.0001%
Playlists (combinatorial)	0.003%

Table 1: Varying Densities for the derived datasets

3.1 Reconstructing Session Data

We define a session (consecutive plays of tracks t by a user u) as $s_{u,m} = \{t_1, \ldots, t_{T_s}\} \in \mathcal{S}^u$ with m many sessions for user u. If the session $s_{u,m}$ is obvious from the context, we omit the subscripts for readability. The tracks in one session are ordered w.r.t. $t_i \prec t_{i+1} \equiv \text{abs}(\text{end}(t_i) - \text{start}(t_{i+1})) < g \quad \forall i = 1, \ldots, T_s - 1$, some $g \geq 0$ and start(·) and end(·) being the start and end timestamp from the logfile, respectivly.

No explicit session information is available in the data. To derive session data, we empirically choose some $g > 0$ based on the lag distribution in the log file.

The lag between two songs is assumed to be (close to) 0, if two songs are enqueued in a playlist and played after the other. This, however, is not always true, as depicted in Figure 1.

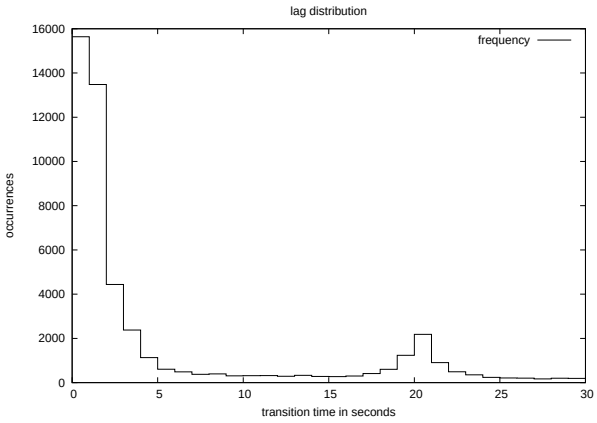

Figure 1: Gap between consecutive songs

Two effects can easily be seen in the chart: First, there are several lags close to 0 seconds. Second, there is a small increase around the lag of 20 seconds. We interpret lags close to 0 to come from the data collection mechanism, that is, log entries are rounded to the nearest second, and the devices need some time to (pre)load the next song to play, or cross-fades are used. Gaps around the duration of 20 seconds are assumed to define 'end-of-playlist'-markers, that is, the last song within a playlist ends, the user recognises 'silence' after a short amount of time, locates his device, and loads/starts another playlist. Since there is no indication that a consecutive playlist is thematically close to the

[3] http://www.zune.net

one before, we assume $g = 10$ seconds for session splitting.

Doing so we are following standard literature: E.g., in [5], track transitions are mined from online radio stations (possibly also containing advertisements). They use lags of 20 seconds. In [7], it is assumed that a session breaks if the elapsed time between two songs exceeds 5 minutes.

Using this technique, we yield 8.8 million sessions from the 18.4 million log entries. As it will become clear in Section 4, we only use those sessions having at least two tracks. We also require each user to contribute at least two sessions. This way we can avoid the 'new user' problem.

Figure 2 shows the occurrences of different session lengths in our data (See the 'all-pair' line there).

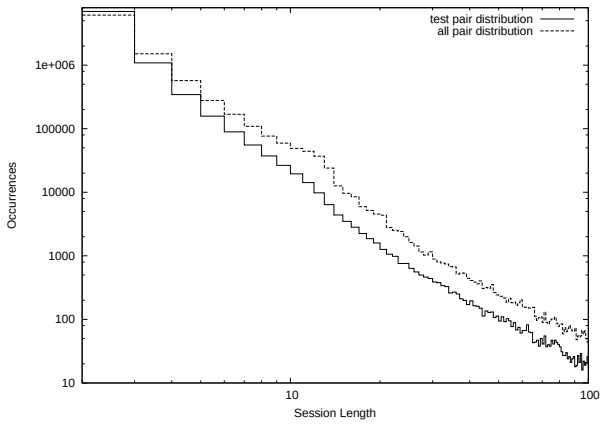

Figure 2: Distribution of session lengths

3.2 Baselines

For comparison with the later presented algorithms, we use the following 'most popular' (MP) baseline methods for next song prediction:

- most often downloaded/purchased songs,
- most often played songs
- most popular by user

Our 'most popular by user' baseline predicts those tracks to the user that he has listened to most. The baseline methods are only trained on the playlog, do not consider sequence information available in the logfile and always output a constant prediction.

3.3 Markov-based prediction

We use a Markov-Chain model of length 1 to incorporate sequence information in the prediction task. We define the probability of the occurrence of the song t_{i+1}, given track t_i, as

$$\hat{p}(t_{i+1}|t_i) = \frac{\mid \{(t'_i, t'_{i+1}) \in \mathcal{S} | t'_i = t_i, t'_{i+1} = t_{i+1}\} \mid}{\mid \{(t'_i, t'_{i+1}) \in \mathcal{S} | t'_i = t_i\} \mid}$$

with $\mathcal{S} = \cup_{u \in \mathcal{U}} \mathcal{S}^u$ when the algorithm is trained on 'Playlog' and When trained on the two playlists variants, $\mathcal{S}$ becomes the union of all transitions derived from the playlists, in either the sequential, or the combinatorial way as described in section 3.

Taking Top-N predictions into account, we define our classifier as

$$\hat{t}_{i+1}(t_i, N) := \arg\max^N_{t \in \mathcal{T}} \hat{p}(t_{i+1} = t | t_i)$$

with a slight abuse of notation as we let arg max return a (ranked) list of the N most probable tracks. Ties in $\hat{p}(t_{i+1}|t_i)$ are broken arbitrarily.

4 Evaluation

4.1 Evaluation protocol

We splitted our data using the 'leave-one-session-out' paradigm. The main idea is to predict either the next or all following songs, given some start/seed songs in that session. In an application scenario, predicting the next song means that we are able to reconstruct the listening history of the user under changing conditions (the recommendation is adjusted each time a next song is played). By predicting multiple following songs, we aim at measuring whether we are able to capture the 'mood' or 'style' of the users' listening preferences for the current session.

In both protocols, we average results over each session $s_m \in \mathcal{S}$ (each having T_s many tracks) by choosing some 'test song' t_j with $j+1 \in [1, T_s - 1]$. Training is done using all other sessions $\mathcal{S} \setminus \{s_m\}$, and songs $\{t_1, \ldots, t_j\} \in s_m$.

Using this technique yielded 4.5 million training sequences (consecutive song plays) and 8.8 million test sequences. Please note that we have more test sequences than training sequences. This necessarily is the case, since we required j to be in $[1, T_s - 1]$, and thus, considering a session of length 2 (our minimal required length), we always use song t_1 as a seed for the recommendation algorithm to predict song t_2.

Figure 2 shows that the test set distribution ('test-pair': the distribution of $j + 1$ plotted on x-axis) reflects the true session length distribution of the session data extracted from the log file. The reason for having more test sequences *of length 2* than actual sessions of length 2 is that each session of length greater than 2 can possibly be reduced to create a test pair for $j = 1$.

Predicting the next song

The first evaluation focuses on next-song prediction, i.e. only song t_{j+1} should be predicted by our algorithm. To evaluate the performance, we calculate the precision of the first N recommended tracks

$$P@N := \text{avg}_{s_m \in \mathcal{S}} I(t^{s_m}_{j+1} \in \hat{t}_{j+1}(t_j, N))$$

with $I(\cdot)$ being 1 if the argument is true, otherwise 0. The precision calculates whether one of the n recommended tracks matches the true following track.

Top-N List intersection

When evaluating against all following songs of a session, we use Top-N List intersection as evaluation measure. It is able to capture whether our recommendation algorithm correctly identifies the 'mood' of the user. It evaluates whether $\hat{t}_{j+1}(t_j)$ matches for some $o \geq j + 1$. Thus, the measure can also be thought of whether our algorithms are able to identify songs a user likes, or dislikes, given the current session/'mood'. Its definition measures the overlap of the recommendations with all following songs within a session and is given by

$$list(N) := \sum_{s_m \in \mathcal{S}} \frac{\mid \hat{t}_{j+1}(t_j, N) \cap \{t_o \in s_m : o \geq j + 1\} \mid}{min(\mid \hat{t}_{j+1}(t_j, N) \mid, T_s - j, N)},$$

again with $N = 5$.

Please note that both lists in the measure may contain less than 5 tracks. Possible reasons are recommendation of less than 5 tracks ($|\hat{t}_{j+1}(t_j, N)| < 5$), or too few tracks to the end of the session ($T_s - j < 5$) in the test data.

4.2 Results

Results for our experiments are given in Table 2.

Algorithm	P@5	list(5)
Markov$_{global}$	43.9	61.8
Markov$_{playlists_sequential}$	11.5	17.6
Markov$_{playlists_combinatorial}$	14.0	23.9
MP$_{user}$	3.2	7.9
MP$_{download}$	1.2	1.8
MP$_{playcount}$	1.2	1.8

Table 2: Results for P@5 and list(5) measure in %

A closer look at the predictions for MP$_{download}$ and MP$_{playcount}$ revealed that the recommended tracks are identical. We consider our filtering process as described in Section 3 as being too restrictive. Furthermore, note that the most popular by user results are not further ranked: A user having heard each of his tracks only once introduces ties which can not be broken in a smart fashion.

What is more interesting is a comparison between our Markov-based recommendations. For next song prediction, the global variant taking collaborative information into account clearly exceeds both variants being trained on the predefined playlists. This is supported from the density estimations given in Table 1, since playlists contain much less sequential information than the log files.

The overall increase of performance for the combinatorial playlist variant over the sequential one for both evaluation measures show that playlists within our data are used to group tracks, rather to reflect sequential transitions.

Having a closer look at the performance of the Markov$_{global}$ recommender broken down by session lengths gives interesting insights to the data, as shown in Figure 3.

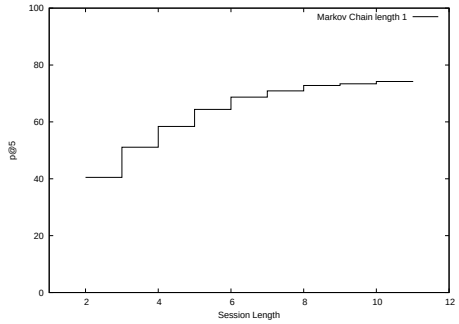

Figure 3: Results for different session length (j) for Markov$_{global}$.

As expected from a collaborative model, performance increases when the available historic data in the sessions also increases. While we have shown in Figure 2 that the total amount of test data decreases for increasing j, the overall trend is still surprising, since we have not considered Markov-lengths > 1 here. We assume that the derived sessions being longer than 2 are based on random plays of only a few tracks, i.e. repetitive sequences are present among different sessions.

5 Conclusion and Future Work

In this work we have presented our initial algorithm to recommend songs to the user to listen to next. We showed that simple collaborative information greatly improves performance compared to baseline methods.

As it has been shown in literature [2][5], incorporating actual content data might help in the recommendation task. However, up to our best knowledge, there is no clear and obvious way on how to do so.

Furthermore, we currently limited our research to only use implicit metadata derived from listening history data, rather than using, e.g., user-defined tags to guide the recommendation process. Further gathering of metadata, be it either content or tag data, will result in the need to define (multiple) similarity measures, which need to be combined in some smart way to improve the recommendation process. Initial studies in this area have been done, e.g., in [5]. However, while their approach shows that learning and using such similarity measures can help, they also implicitly show that current content-based solutions have lots of space for further improvements. To this end, we aim at adding more metadata to our system.

References

[1] Adomavicius G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on Knowledge and Data Engineering, Vol 17, No 6, June 2005 pp. 734-749

[2] Casey, M. A., et al.: Content-Based Music Information Retrieval: Current directions and Future Challenges, Proceedings of the IEEE, Vol 96, No. 4, April 2008, pp. 668-696.

[3] Flexer, A., et al.: Playlist Generation Using Start and End Songs, ISMIR 2008, pp. 173-178.

[4] Logan, B.: Mel Frequency Cepstral Coefficients for Music Modelling, ISMIR 2000.

[5] Maillet, F., et al.: Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists, ISMIR 2009, pp. 345-350.

[6] Nanopoulos A., Krohn-Grimberghe A.: Recommending in Social Tagging Systems based on Kernelized Multiway Analysis, IFCS 2009.

[7] Ragno, R., Burges, CJC, Herley, C.: Inferring similarity between music objects with application to playlist generation, ACM MIR 2005, pp. 73-80.

[8] Schein et al.: Methods and metrics for cold-start recommendations, SIGIR 2002, pp. 253-260.

[9] Sordo et al.: The Quest for Musical Genres: Do the Experts and the Wisdom of the Crowds Agree?, ISMIR 2008, pp. 255-260.

[10] Turnbull, D., Barrington, L., Lanckriet, G.: Five Approaches to Collecting Tags for Music, ISMIR 2008, pp. 225-230.

[11] Yang, B., et al.: Online video recommendation based on multimodal fusion and relevance feedback, ACM CIVR 2007, pp. 73-w80.