Embedding-based next song recommendation for playlists

Raphaël Romero[1] and Tijl De Bie[1]

1- Ghent University - Department of Electrical Engineering
Technologiepark-Zwijnaarde 126, 9052 Gent - Belgium

Abstract. In recent years, music storage and consumption has shifted massively to digital platforms, where large-scale libraries of songs are stored along with their metadata. As a byproduct of this transformation, music is increasingly being organized and accessed in the form of playlists. User-curated playlists have become massively available online, and the challenge of automatically generating playlists has gained popularity in the music information retrieval community. In this paper, we build on link prediction for graphs to propose a flexible music playlist generation method. We transform a playlist dataset into a weighted graph of songs and posit a Poisson model on the count of transitions between songs, where the rate is modulated by the euclidean distance between song embeddings. Our method yields prediction results superior to common deterministic baselines, suggesting that the learned embeddings can be used to derive a meaningful notion of song similarity.

1 Introduction

Playlists are omnipresent on streaming platforms nowadays, and represent a unique source of data for learning similarities between songs in music databases. Thus, their empirical study has lead to a substantial body of work in the Music Information Retrieval (MIR) community. A full survey on the different types of playlists and challenges in playlist modelling is provided in [1].

Among these various challenges, *next-song recommendations* consists in scoring each potential next songs given a set of seed songs, or *playlist history*. Naive approaches include for instance scoring the next songs purely based on their popularity in the training set, or based on the similarity of their meta-data and those of the seed songs. While these two methods already yield superior performances than random prediction, the former doesn't enable discovering new/unknown tracks, while the latter would most likely create playlists with limited diversity.

To cope with these limitations, we argue that the notion of score depends heavily on the considered dataset, and that it can be learned from a history of user-curated playlists. To achieve this, we propose to rely on the (multi)graph formed by the transitions between songs in a playlist dataset to further generate vector representations of the songs. The song similarities can then be measured through the Euclidean distance between their latent representations.

Contributions We cast the next song prediction into a weighted graph embedding based link prediction problem. We generalize a graph embedding method designed originally for unweighted graphs, and apply it to the next-song recommendation task on playlists. Furthermore, we evaluate our method compared to effective deterministic baselines, with results suggesting that our embeddings are able to learn meaningful songs representations.

Related Work Playlist prediction is a popular topic in the MIR community. For instance playlist continuation has been the subject of a recent challenge [2]. Moreover, modelling playlists as random walks on graphs is an already popular approach studied for instance in [3] or [4]. Similar to our approach, the latter is an example of an approach relying on a notion of distance in a latent space to define the song similarities. However, to our knowledge none of these previous approaches directly embed the weighted transition count graph to generate song representations.

Outline In section 2 we introduce the problem setting and our method to solve it. In section 3 we describe our evaluation setup and discuss the results.

2 Problem statement and proposed method

In this section we introduce formally the next-song recommendation problem, and present our approach, based on a weighted graph connecting the songs in the database based on number of transitions between each song pair in a training dataset.

2.1 Playlist prediction

As there are many ways to define the challenge of playlist prediction, in this section we describe the specific problem of next song prediction.

Let $\mathcal{S}$ be a set of songs, indexed for instance by integers. We define a playlist as an ordered set of songs $p = (s_1, ..., s_K)$, where K is the length of the playlist and $s_j \in \mathcal{S}$ are songs. A playlist dataset is a collection of N playlists: $\mathcal{D} = (p_1, ..., p_N)$. The task we wish to solve is defined as follows. For each playlist $p = (s_1, ..., s_K)$, we wish to predict the $k-th$ song of the playlist given the $k-1$ first songs. We suppose that each song s_k is chosen among a set of candidate songs $C(s_{1:k-1})$ that depends on the history $s_{1:k-1}$. Then we define our next song prediction task as a ranking task, where each song $s \in C(s_{1:k-1})$ is ranked given the history based on a score $f(s; s_{1:k-1})$.

2.2 From playlist data to a weighted song graph

In order to derive a song similarity based on the playlists, we calculate a multigraph of songs, where each song is represented by a node, and two given nodes form a link every time a transition between them appears in the training set. We define the adjacency matrix $W = (w_{ij})$ of this graph as the symmetric matrix

such that for two songs i and j, w_{ij} is the number of transitions between i and j.

2.3 Our method: encoding the latent structure of the song graph into embeddings

In the following we propose a latent variable model for the transition counts, and describe our approach to solve for the maximum-likelihood estimates of these latent variables.

2.3.1 Weighted Graph embedding

Network embedding, also known as graph representation learning, is a class of methods for mapping the nodes in a graph to a Euclidean space, such that the vector representations of the nodes reflect structural properties of the graph.

Recently, a probabilistic approach for graph embedding has been proposed [5]. In this framework, the Bernoulli probability for a node pair (i, j) of being connected is modelled as a decreasing function of the distance between their associated vector representations. As a consequence, maximizing the likelihood of an observed graph with respect to the embeddings will enforce similar songs to have nearby embeddings in the latent space. Furthermore, this framework allows including prior information about the topology of the graph, in order to factor out this information from the learned embeddings.

While the existing method doesn't generalize flexibly to weighted cases, here we adapt this method by modelling the weights between pairs of nodes independently as Poisson distributed, with rate parameters expressed as a decreasing function of the distance between their corresponding embeddings.

Mathematically, for two nodes i and j we suppose that the number of transitions between them, w_{ij} is distributed as:

$$\begin{aligned} w_{ij} &\sim Poisson(\lambda_{ij}) \\ \lambda_{ij} &= exp(b_{ij} - d(z_i, z_j)) \end{aligned} \tag{1}$$

where z_i,z_j are respectively the embeddings of song i and song j, b_{ij} is a pair-wise bias term and $d(z_i, z_j)$ is the Euclidean distance between them.

The bias term b_{ij} can be used to incorporate prior notions of similarity between the songs, such that this information needs not be reflected in the Euclidean distance between the latent embeddings.

Remark While we present a simple undirected version of our model, other variants can be proposed. For instance the Poisson distribution proposed here for pairwise count data can be replaced by any exponential family of distribution, in order to model pairwise response variable defined in other domains. The exponential non-linearity/ activation function used here, can also be modified accordingly depending on the type of data and distribution. Finally, the directionality of the playlists can be modelled by duplicating the embeddings, in order to obtain an asymmetric array of parameters such that $\lambda_{ij} \neq \lambda_{ji}$.

2.3.2 Parameter estimation

As our goal is to use the above model mainly for prediction, we propose to solve for the Maximum Likelihood estimates of the latent variables $(z_i)_{i \in \mathcal{S}}$, without considering the level of uncertainty of the estimates. This yields an unconstrained, non-convex optimization problem where the loss function is given by the negative log-likelihood of the model 1 and writes (using the same notations):

$$L(z) = \sum_{i \neq j} \lambda_{ij} - w_{ij} log(\lambda_{ij})$$

To circumvent the $O(|S|^2)$ complexity required to evaluate this loss function, we propose to approximate it by sub-sampling for each positive edge a number ρ of negative edges:

$$L(z) \approx \sum_{i} \sum_{j \in N(i) \cup \tilde{N}(i)} \lambda_{ij} - w_{ij} log(\lambda_{ij}).$$

Where for each node i, $N(i)$ denotes its set of neighbors, $\bar{N}(i) \subset S \backslash N(i)$ denotes a randomly sampled set of negative neighbors, of size$|N(i)| \times \rho$, where ρ is the negative/positive sample ration. To solve that optimization problem we use the Adam algorithm [6] with a learning rate of 10^{-2}. At each iteration of the algorithm, we resample a set of negative neighbors for each node.

3 Evaluation

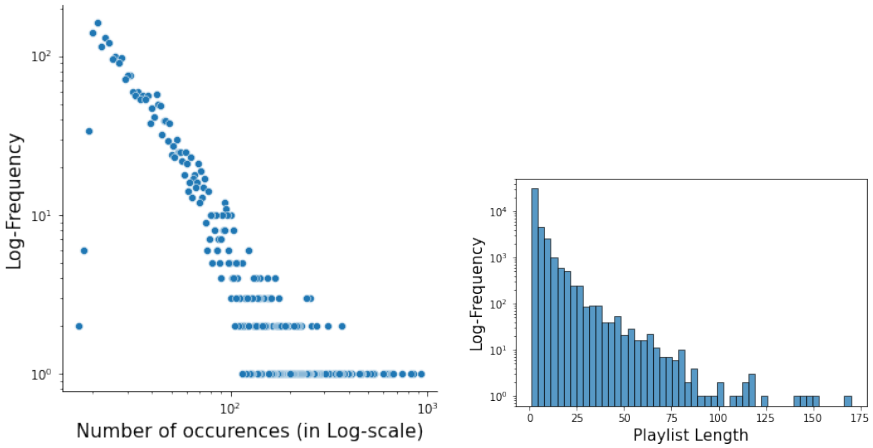
In other to assess the suitability of the proposed graph embedding method for the specific problem of next-song prediction, in this section we introduce our evaluation setup. Then we describe the results obtained on a real-world playlist dataset.

3.1 Data

We evaluate our approach on the Yes.com playlist dataset provided in the paper [4]. The distribution of popularities of the songs in the dataset as well as the distribution of playlist lengths are shown in figure 1. In our experiments we use the full training set of 41480 playlists as a training set, and randomly sample a set of 500 playlists among the 389728 provided, to use as a test set.

3.2 Baseline methods

We describe several baseline methods used in our experiments. The **Popularity (Pop)** baseline just ranks the possible next songs according to their number of occurrences in the train playlists. The **Co-occurrence (Cooc)** ranks the possible next song according to their number of co-occurrences with the current



(a) Log-log plot of the song occurrences (b) Distribution of the playlist lengths

Fig. 1: Statistics of the Yes.com dataset

song in the train playlist. The **Meta-Data (MetaData)** baseline ranks the possible next songs based on the Jaccard index of their set of tags and the ones from the current song. We compare all these baselines to our proposed Poisson Embedding (**PoissonEmb**) method.

3.3 Results and Discussion

Method	Pop	Cooc	MetaData	PoissonEmb
MRR	0.547 ± 0.017	0.730 ± 0.013	0.536 ± 0.013	$\mathbf{0.815 \pm 0.007}$
Recall@5	0.745 ± 0.015	0.747 ± 0.014	0.731 ± 0.016	$\mathbf{0.979 \pm 0.003}$

Table 1: Results

To evaluate our method, we adapt the evaluation procedure proposed in [7]. Let's consider a test playlist $(s_1, ..., s_K)$. For each transition $s_{k-1} \rightarrow s_k$ in the playlist we calculate a set of candidate songs, denoted $C(s_{1:k-1})$, such that the true next song s_k is in $C(s_{1:k-1})$. Next we compute a score $f(s'; s_{1:k-1})$ for each candidate song $s' \in C(s_{1:k-1})$, base on the different methods. Finally, we rank the *true next song* s_k based on this obtained list of scores. To form the set of candidate songs, we propose to use $C(s_{1:k-1}) = \{s_k\} \cup C'$, where C' is a set of songs of fixed size $N_{candidates}$, sampled at random among the songs not in $s_{1:k-1}$ (i.e. not featured yet in the playlist).

Based on the obtained list of ranks we report two metrics, namely the mean reciprocal rank (MRR), and the recall at K (Recall@K) defined as the proportion of ranks that are below K. In our experiments we use 10 negative candidate songs for each transition and report the recall with K=5.

For our proposed Poisson embedding method, we used 8-dimensional embeddings, and train the model by using 1 negative neighbor per positive ones. Moreover, as ranking only depends on the set of euclidean distances between the embeddings, we set all the bias terms to zero.

As the test score is stochastic due to the presence of random candidates, we repeat the evaluation 10 times and report the mean and variance of the metrics. The results are shown in table 1. Among the baselines, we can see that just using the co-occurrence already allows to recover a large portion of the next songs into the top 5. Moreover, the Meta-Data-based baseline does not perform better than the Popularity, confirming a systematic bias of playlists towards popular songs. However, our proposed method yields superior results that these deterministic baselines, suggesting that the learned embeddings are successful in learning an abstract notion of song similarity.

4 Conclusion

In the present work we have proposed a graph embedding based method to learn song similarities based purely on a history of user-curated playlists. While conceptually simple and flexible, this method allows learning a similarity function that perform better than common baselines in our offline next song prediction experiments. While in our approach, the playlist dataset is first summarized into a transition count graph, other summarizing statistics could be considered, while modifying the exponential family in the model 1 accordingly, following Generalized Linear Models routines.

Acknowledgements The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) (ERC Grant Agreement no. 615517), and under the European Union's Horizon 2020 research and innovation programme (ERC Grant Agreement no. 963924), from the Special Research Fund (BOF) of Ghent University (BOF20/IBF/117), from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme, and from the FWO (Fonds Wetenschappelijk Onderzoek , project no. G0F9816N, 3G042220).

References

[1] Geoffray Bonnin and Dietmar Jannach. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, 47(2), 2014.

[2] Ching Wei Chen, Markus Schedl, Paul Lamere, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. *RecSys 2018 - 12th ACM Conference on Recommender Systems*, (1):527–528, 2018.

[3] Brian McFee and Gert Lanckriet. Hypergraph models of playlist dialects. *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, (Ismir):343–348, 2012.

[4] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 714–722, 2012.

[5] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Conditional Network Embeddings. *7th International Conference on Learning Representations, ICLR 2019*, may 2018.

[6] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[7] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. Order, context and popularity bias in next-song recommendations. *International Journal of Multimedia Information Retrieval*, 8:101–113, 6 2019.