Towards Playlist Generation Algorithms Using RNNs Trained on Within-Track Transitions[*]

Keunwoo Choi
Centre for Digital Music
EECS, QMUL
London, UK
keunwoo.choi@qmul.ac.uk

György Fazekas
Centre for Digital Music
EECS, QMUL
London, UK
g.fazekas@qmul.ac.uk

Mark Sandler
Centre for Digital Music
EECS, QMUL
London, UK
m.sandler@qmul.ac.uk

ABSTRACT

We introduce a novel playlist generation algorithm that focuses on the quality of transitions using a recurrent neural network (RNN). The proposed model assumes that optimal transitions between tracks can be modelled and predicted by internal transitions within music tracks. We introduce modelling sequences of high-level music descriptors using RNNs and discuss an experiment involving different similarity functions, where the sequences are provided by a musical structural analysis algorithm. Qualitative observations show that the proposed approach can effectively model transitions of music tracks in playlists.

CCS Concepts

•Computing methodologies → Neural networks; **•Applied computing → Sound and music computing;**

Keywords

recurrent neural networks; music playlists; music recommendation

1. INTRODUCTION

In music recommendation, the quality of transitions become important particularly when the recommendation is provided in the form of a playlist. This is due to a unique aspect of music consumption. Unlike other products, music is consumed *i) instantaneously*, for instance, while listening using streaming services, *ii) repeatedly*, i.e., listeners are willing to listen to the same music multiple times, and *iii) quickly*, i.e., an item usually only lasts a few minutes. Hence, recommended items are typically consumed or played in a sequence. This behaviour introduces the need for *good transitions* between items, that is, the relevance and subjective judgement a recommended track depends on the previous track.

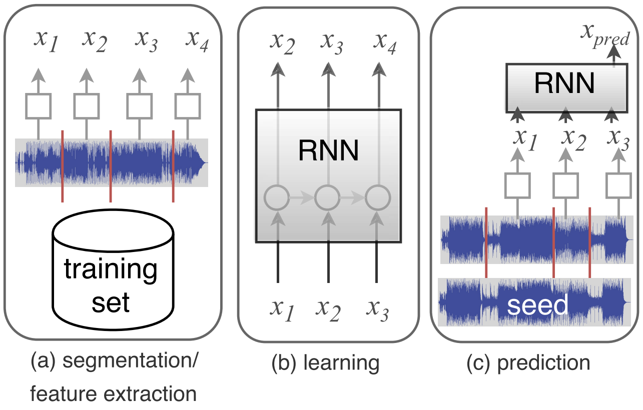
Recommendation approaches using collaborative filtering are prone to overlook niche or new items, although the popularity bias of known items can be compensated for. This is called the *cold-start problem* [20]. Content-based approaches which are designed to solve the cold-start problem can suffer from lack of diversity when recommended items are selected simply by similarity. This is often called top-N recommendation. It is well known that *unexpectedness*, *surprise* or *serendipity* play an important role in the music recommendation and discovery [4]. Compared to other strategies, focusing on transitions can naturally provide these qualities.

There have been approaches that primarily focus on the transitions of tracks [13], [3], [15]. They assumed the *Markov* property of hidden states or embeddings of tracks. Using the Markov property, it is assumed that future events only depend on the current one and does not depend on the past. This has been successfully used for sequence modelling for instance in speech [19] too. In music computing, playlist datasets [16], [14], [3] collaboratively created for reference by DJs and listeners were used for training and evaluation of sequence modelling approaches. Although these datasets consist of a large number of tracks, e.g. 101k playlists in [16], the lack of audio data fundamentally limits research based on audio content analysis.

Recently, recurrent neural networks (RNNs) have become widely used for sequence modelling in tasks such as speech recognition, substantially outperforming previous hidden Markov model-based approaches [22]. The success of the application of RNNs largely relies on the introduction of Long Short-Term Memory (LSTM) units [10]. The merit of LSTM comes from the gate cells of LSTM units, that decide how much the units take input, release output, and forget the previous events. Especially, the forget gate improves the training efficiency by helping the gradients flow well. However, RNNs have not been used for playlists generation and modelling, due in part to the lack of sufficient training data. To solve this problem, we propose using an RNN trained on *within-track* transitions to model playlists.

We assume that transitions between structural segments of music can be used as a model for generating the desired high-quality transitions between tracks. In general, segments in a track are different but coherent and their

[*]This work was part funded by the "Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption" (FAST IMPACt) EPSRC Grant EP/L019981/1 and the European Commission H2020 research and innovation grant AudioCommons (688382). Sandler acknowledges the support of the Royal Society as a recipient of a Wolfson Research Merit Award.



(a) segmentation/ feature extraction (b) learning (c) prediction

Figure 1: A block diagram of the proposed algorithm, (a,b) training of RNN and (c) prediction of a feature vector, x_{pred}.

musical features can be expected to match well in succession. This is due to the careful and intentional design by the composer. Using this approach, the number of transitions can easily outnumber that of existing playlist datasets, and therefore it enables to train an RNN model.

The rest of the paper is organised as follows. The proposed method is first described in Section 2. We then present experimental results and discussion in Section 3 and conclude in Section 4.

2. THE PROPOSED MODEL

Figure 1 illustrates the procedure of dataset construction, as well as the training and prediction stages of the proposed algorithm. First, the training tracks are segmented and x_i, the features for each segments are extracted (Fig. 1a). Then an RNN of length N (N=3 in the figure) is trained to learn the transitions of the sequence of feature vectors (Fig. 1b). When a seed track is provided, the features of the last N segments are extracted and fed into the trained RNN to predict the feature vector x_{pred} (Fig. 1c). The algorithm selects a track with a start segment that is most similar to x_{pred}.

2.1 Structural Segmentation

Structural segmentation is a task aiming to find the boundaries of different segments or parts in music, e.g. *intro, verse, bridge, chorus*. The most common approach is to take advantage of self-similarity between frames of the track [9]. In the experiment, we used a basic and efficient method that is proposed in [9]. Although the results introduce some errors, the feature vector sequences that are based on the imperfect segmentation still approximate the information about how each feature changes along time in each track.

2.2 Feature Extraction

The proposed algorithm can use feature extraction methods that are relevant to listeners' musical preferences and able to represent a musical segment. This includes estimated latent features from collaborative filtering [26], tags such as genre and emotion [7] or implicit features such as the weights of the last hidden layer of a neural network classifier [12]. Using explicit labels such as genre can facilitate explaining the behaviour of the algorithm, which is important for research and also to the listener.

In the experiment, an auto tagging algorithm in [5] is used to predict a 50-dimensional vector whose elements correspond to the probability of each tag. The tagging algorithm is based on deep convolutional neural networks and trained on Million Song Dataset [2]. It shows state-of-the-art performance while the tags cover variety of categories such as genre, emotion, instrument, and era. Although some of the tags such as genre typically characterise the entire music track, they are not necessarily constant over the whole track.

2.3 RNN Model

The goal of RNN training is to predict the feature of the following track (x_{pred}) that maintains consistency and fluctuations, i.e., a certain variation over the features. To this end, a 2-layer RNN with 512 hidden units is employed. LSTM units [10] are used as they show state-of-the-art performance among RNN variants for several sequence modelling tasks [11].

2.4 Similarity Measure

A similarity measure is necessary to find the subsequent track using the feature vector predicted by the RNN. The similarity metric directly affects the properties of the generated playlists and therefore it should be carefully selected. Using the *cosine distance* may compensate for the popularity bias and result in recommending more niche items [23]. The *Discounted Cumulative Gain* (DCG) turned out to be effective in our experiment.

DCG is a weighted version of *Cumulative Gain* (CG). CG is designed to measure ranking quality of a retrieved list and DCG weights on the top-N relevant items by *discounting* lower relevant items. Applying this measure was motivated by the type of feature extraction algorithm we use. The extracted feature is a vector of probabilities of each tag and tags with large probabilities should be weighted more than the others to facilitate maintaining the consistency of the generated playlists. Because DCG weights high-ranking elements more, it can theoretically work better.

3. RESULTS AND DISCUSSION

3.1 Configurations

For training, we used a private dataset with 28,430 commercial tracks of modern popular music including Rock, Hip-Hop, and Jazz. We used 7,880 tracks from the *ILM10k* dataset for testing [21] [1].

The segmentation is performed using [9], which is implemented in [18]. As mentioned in Section 2, an automatic tagging algorithm was used as a feature extractor [5]. An RNN with a length of 50 is trained. We compared DCG with the cosine distance and the l_2 norm for computing similarity. Audio processing and RNN are implemented using *librosa* [17], *Keras* [6], and *Theano* [25].

Figure 2 shows the transitions of feature vectors in three playlists given the same seed track but different similarity metrics. The seed track is represented by a 7-by-50 matrix, i.e., 7 segments of the track. White horizontal lines indicate

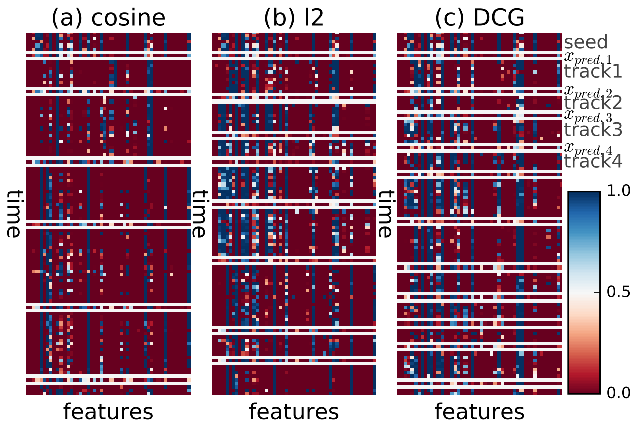


Figure 2: Transitions of feature vectors by cosine distance, l_2, and DCG on left, centre, and right, respectively. Y-axis is time (top to bottom) and x-axis refers to the 50 feature dimensions. The first track (topmost feature vectors) is the seed.

beginnings and ends of each track. The predicted feature vectors (1-by-50) are illustrated in between.

The figure helps to explain several aspects of qualitative observations by the authors while listening to the generated playlists.

3.2 Discussions

First, we found both consistency and fluctuations in the extracted features within tracks. In general, several features show consistently large (blue) and small (red) values, while the other features vary. It supports the selection of feature extraction algorithm.

However, there are still rooms for further improvements. For example, whitening of each feature dimension can be adopted to compensate the prior distributions of each feature. Although RNNs are able to adapt to such differences, the similarity measure may be affected from such pre-processing.

Second, the transitions usually successfully keep the coherence within playlists as demonstrated by the figure. However, we noticed that the model is prone to missing overall similarity in long playlists. It may be related to the observation that the trained RNN occasionally predicts a vector that does not have near neighbours. It means the selected track is not similar enough to the predicted vector, which may result in an undesirable or suboptimal transition.

This may first be due to the short lengths of segments in the training data. The majority of training tracks have fewer segments (90% of the training tracks has less than 17 segments), therefore the long-term dependency may not be learnt and the prediction may be dominated by short-term features.

In future work, training with longer sequences such as concatenated features of tracks from sequences in an albums, setlists or curated playlists may be used to help learning better transitions. It can be also a typical behaviour of RNNs. Although RNNs generally model the long-term dependency of sequences, in many cases, RNNs have shown a behaviour puts more emphasis on recent inputs rather than older ones.

However, the problem seems to be partly resolved when DCG is used as the similarity measure as discussed alongside out last set of observations.

Third, DCG provides more coherent playlists compared to the cosine distance and l_2-norm. This phenomenon is found not only in the sequences corresponding to Figure 2 but also in other playlists. This can be explained as follows. In each track, there are consistently strong, consistently weak, and fluctuating features. This pattern, especially the consistencies, can be easily learned by the RNN and the consistent features are maintained in the predictions. Finally, DCG prioritises the features that are large in predictions, resulting in successfully finding a track with those consistently large features. This improves the coherence of the resulting playlists.

4. CONCLUSION

In this paper, we proposed a novel algorithm for playlist generation that relies on learning desirable musical qualities from within-track transitions between musical segments. The proposed combination of an RNN, within-track structure and DCG showed an encouraging result. Within-track structures showed the consistency and dynamics that are assumed in playlists. An RNN model learnt the feature sequences and its predictions are successfully used for the selections of following tracks. Different similarity measures resulted in different playlists.

Future work will investigate advanced architectures such as bidirectional RNNs [24] and more formal assessments. Using bidirectional RNNs can be used to create playlists that have more constraints e.g. start and end tracks [8] and steerability [14]. Formal assessments will include subjective measurement e.g. satisfaction and objective measures with regards to consistency, fluctuations and diversity.

5. REFERENCES

[1] Allik, A., Fazekas, G., Barthet, M., and Saldler, M. myMoodplay: an Interactive Mood-based Music Discovery App. In *Proc. of the 2nd Web Audio Conference (WAC), April 4-6, 2016, Atlanta, Georgia.* (2016).

[2] Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011* (2011), pp. 591–596.

[3] Chen, S., Moore, J. L., Turnbull, D., and Joachims, T. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), ACM.

[4] Choi, K., Fazekas, G., and Sandler, M. Understanding music playlists. In *Proceedings of the ICML 2015: Machine Learning for Music Discovery Workshop* (2015), ICML.

[5] Choi, K., Fazekas, G., and Sandler, M. Automatic tagging using deep convolutional neural networks. In *Proceedings of the Conference on*

International Society of Music Information Retrieval, New York, USA (2016), ISMIR.

[6] Chollet, F. Keras: Deep learning library for theano and tensorflow. https://github.com/fchollet/keras, 2015.

[7] Dieleman, S., and Schrauwen, B. End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (2014), IEEE, pp. 6964–6968.

[8] Flexer, A., Schnitzer, D., Gasser, M., and Widmer, G. Playlist generation using start and end songs. In *ISMIR* (2008), pp. 173–178.

[9] Foote, J. Automatic audio segmentation using a measure of audio novelty. In *IEEE International Conference on Multimedia and Expo, (ICME) 2000.* (2000), IEEE.

[10] Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm. *Neural computation 12*, 10 (2000), 2451–2471.

[11] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069* (2015).

[12] Liang, D., Zhan, M., and Ellis, E. D. P. Content-aware collaborative music recommendation using pre-trained neural networks. In *Proceedings of the Conference on International Society of Music Information Retrieval* (2015), ISMIR.

[13] Liebman, E., Saar-Tsechansky, M., and Stone, P. Dj-mc: A reinforcement-learning agent for music playlist recommendation. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems* (2015).

[14] Maillet, F., Eck, D., Desjardins, G., Lamere, P., et al. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the Conference on International Society of Music Information Retrieval* (2009), ISMIR, pp. 345–350.

[15] McFee, B., and Lanckriet, G. R. The natural language of playlists. In *Proceedings of the Conference on International Society of Music Information Retrieval* (2011), ISMIR, pp. 537–542.

[16] McFee, B., and Lanckriet, G. R. Hypergraph models of playlist dialects. In *Proceedings of the Conference on International Society of Music Information Retrieval* (2012), ISMIR.

[17] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference* (2015).

[18] Nieto, O., and Bello, J. P. Msaf: Music structure analysis framework. In *Proceedings of the Conference on International Society of Music Information Retrieval* (2015), ISMIR.

[19] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[20] Ricci, F., Rokach, L., and Shapira, B. *Introduction to recommender systems handbook.* Springer, 2011.

[21] Saari, P., Fazekas, G., Eerola, T., Barthet, M., Lartillot, O., and Sandler, M. Genre-adaptive semantic computing and audio-based modelling for music mood annotation. *IEEE Transactions on Affective Computing (TAC)* (2015), 1–17.

[22] Sak, H., Senior, A., and Beaufays, F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128* (2014).

[23] Schedl, M., Knees, P., McFee, B., Bogdanov, D., and Kaminskas, M. Music recommender systems. In *Recommender Systems Handbook.* Springer, 2015, pp. 453–492.

[24] Schuster, M., and Paliwal, K. K. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on 45*, 11 (1997), 2673–2681.

[25] Team, T. T. D., Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688* (2016).

[26] Van den Oord, A., Dieleman, S., and Schrauwen, B. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems* (2013), pp. 2643–2651.