Spotify ACM RecSys Challenge: An Embeddings Approach

Liam Byrne
31679498
lhb1g20@soton.ac.uk

Samuel Watson
31662129
sw9g20@soton.ac.uk

Joseph Padden
31588085
jp3g20@soton.ac.uk

Ben Hipwell
30983592
bh3g19@soton.ac.uk

Abstract

*Music playlist continuation remains a key challenge in recommendation systems for music streaming services in 2024. This project revisits the 2018 ACM RecSys challenge and applies the latest embedding representation methods for representing tracks (**Track2Vec**), artists (**Artist2Vec**), and playlist titles. Three models with increasing complexity were developed for the downstream task of playlist continuation. Notably, a Track2Vec mean-pooling approach achieved **58th out of 253 teams on the global leaderboard** which validates the embedding approach for playlist continuation and signposts for future development.*

1. Introduction

The 2018 ACM RecSys Challenge [1] is hosted by Spotify and is centred on *playlist continuation*. This involves recommending additional tracks for users to include in their playlists. The most successful solutions typically used either Matrix Factorization [2, 3] to approximate the user-item interaction matrix or Auto-Encoders [4, 5] to learn low-dimensional representations of playlists [6]. This paper revisits the challenge and applies recent representation learning techniques such as static [7, 8] and transformer-based [9] embeddings for representing tracks, artists, and playlist names.

1.1. Dataset

Table 1 provides a breakdown of the 1 Million song dataset. The dataset was inserted into an SQLite database with 5 tables (track, album, artist, playlist, playlist_track) to efficiently execute queries.

Table 1. Dataset summary

	Count	Mean per playlist
Playlists	1,000,000	-
Unique Tracks	2,262,292	65.465
Unique Artists	295,860	37.692
Unique Albums	734,684	49.597

2. Representation Learning

2.1. Track2Vec

Figure 1 illustrates the architecture used to generate dense embedding representations of tracks. *Track2Vec* employs a Continuous Bag of Words (CBOW) to learn a low-dimensional representation of each track [8, 7]. CBOW is commonly applied for learning word embeddings such as Word2Vec which trains a model to predict a masked *target* word based on the surrounding *context* words. In Track2Vec this is applied over the sequence of tracks in a playlist. This is under the hypothesis that there is a level of **locality** of tracks within the playlist that can be captured by an embedding representation. To constrain the dimensionality of the layers, embeddings are only generated for tracks with 50+ appearances across all playlists (5.19% of original songs). Track2Vec adopts the Negative Log likelihood (NLL, Equation 1) loss used in the original Word2Vec implementation [7].

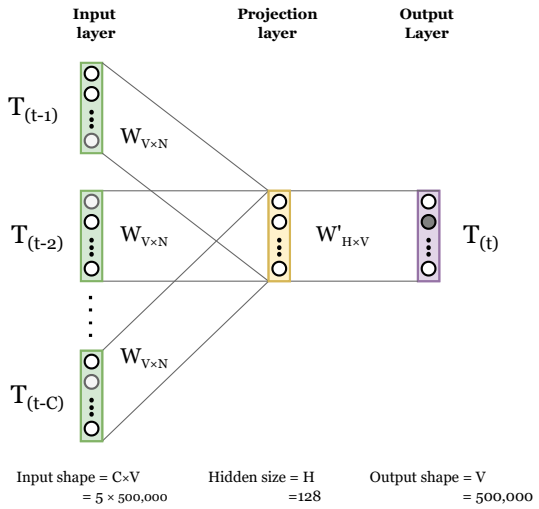

Figure 1. Continuous Bag of Words architecture [8] for building Track2Vec embeddings. This diagram was created internally by our team.

$$NLL = -\sum_{i=1}^{n}(y_i \log \hat{y} + (1 - y_i) \log(1 - \hat{y}) \qquad (1)$$

Table 2. Characteristics of Track2Vec Embeddings

	Track2Vec
Vocab size	117,487
Dimensions	100
Context size	5
Dataset size	1M playlists
Epochs	5

2.2. Artist2Vec

Incorporating the artists into the prediction pipeline was a key consideration of the 2nd place submission [5], as it brings deeper insights into the music genre. As depicted in Figure 2, *Artist2Vec* adopts a skip-gram architecture. Contrary to CBOW, skip-gram predicts the surrounding context tracks. This choice was influenced by Mikolov *et al.* [7] that demonstrated skip-gram being more effective at learning representations of less-frequent instances This was seen as beneficial for representing the $\approx$ 300k unique artists in the dataset, roughly 2x the vocabulary size of the English language.

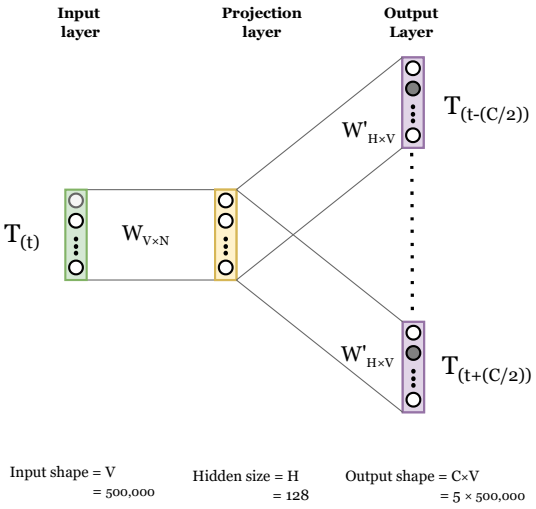

Figure 2. Skip-gram architecture [7] for building Artist2Vec embeddings. This diagram was created internally by our team.

Table 3. Characteristics of Artist2Vec Embeddings

	Artist2Vec
Vocab size	261k
Dimensions	64
Context size	5
Dataset size	1M playlists
Epochs	5

2.3. Representing Playlist Names

The dataset also provides user-generated playlist names. This may capture additional context for playlist continuation. Transformer encoders such as BERT [10] have been shown to be highly effective for learning representations of text. BERT is trained on a Masked Language Modelling (MLM) task to infill masked segments of the word. To represent playlist names, the RoBERTa [11] model was adopted. It has 125M parameters pre-trained on 160GB of web-based text datasets. Figure 3 presents how the pre-trained model will be used to obtain a fixed-size representation. That is, only the hidden representation of the `[CLS]` token is retrieved following guidance by Devlin *et al.* that it can be used as a *sentence-level* embedding.

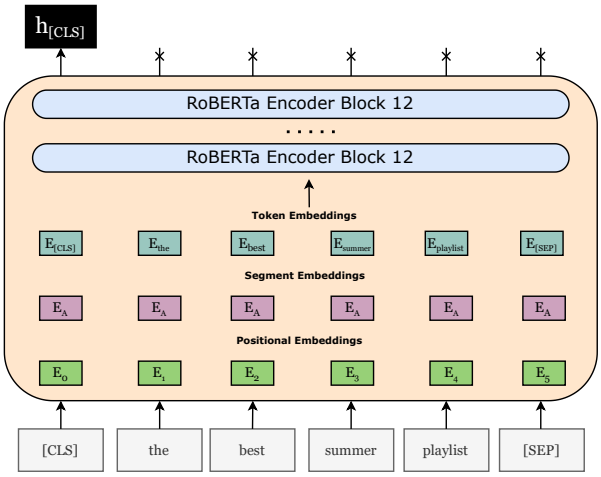

Figure 3. RoBERTa [11] architecture for generating a fixed-size representation of playlist names. This diagram was created internally by our team.

3. Playlist Continuation Model

Now that we have representations for tracks, artists, and playlist names; the downstream task is to combine these embeddings for predicting the next 500 tracks of a given playlist.

3.1. Model 1: Mean Pooling for Track2Vec

This approach first calculates a *playlist embedding* PE by finding the mean of the set of Track2Vec embedding t_i in the playlist P (Equation 2). Next, this *playlist embedding* is used to find 500 of the most similar songs based on the *cosine distance* (Equation 3).

$$PE = \frac{1}{||P||} \sum_{i=1}^{||P||} t_i \tag{2}$$

$$t'_i = \text{argmin}_{\vec{x}} \left(1 - \frac{\vec{PE} \cdot \vec{x}}{\|\vec{PE}\| \|\vec{x}\|} \right) \tag{3}$$

3.2. Model 2: Embedding Enrichment

Rather than solely considering the Track2Vec embedding, the Artist2Vec and playlist name embeddings can be integrated using an embedding enrichment process adopted by Vintch *et al.* [12]. For this dataset, that meant using *Artist2Vec* representations to create average artist embeddings, using RoBERTa `[CLS]` to represent playlist title names and also adding metadata such as the number of albums in the playlist. These values would then be passed into an MLP with ReLU activations with the output being a tensor of size 64 representing an *enriched Track2Vec embedding*. This embedding is then used to find the 500 closest tracks using cosine similarity. To optimize model performance, the loss function is tailored to the evaluation metrics for the competition (Equation 12) [1], thereby guiding the learning process to meet the desired criteria.

3.3. Model 3: LSTM for Embeddings Aggregation

An alternative approach to combining the embeddings is to train a Long Short-Term Memory (LSTM) network over the sequence of track and artist embeddings [13]. Each track embedding is concatenated with the corresponding artist embedding and playlist name embedding. By sequentially passing this through the LSTM, the temporal nature of playlist creation can be captured. This approach is particularly flexible as it can accept varying length playlists. This allows us to train one model that could accept any number of *seed* tracks for playlist continuation.

4. Evaluation

4.1. Quality of Embedding Representations

Figure 4 illustrates a representation of our *Song2Vec* embeddings on two principal components. Whilst there are few clear clusters in this abstraction, clear genre-based grouping offers promise for the downstream utility of these representations.

Table 4 presents the nearest songs based on cosine distance to Taylor Swift's *Love Story*. All of the top 5 nearest tracks are from Taylor Swift, this is a **highly positive indicator of successful clustering**.

Table 4. Tracks with the greatest Track2Vec cosine similarity to Taylor Swift's Love Story

Artist & Track	Cosine Similarity
Taylor Swift: Love Story	**1.000**
Taylor Swift: You Belong With Me	0.914
Taylor Swift: Fifteen	0.883
Taylor Swift: Our Song	0.881
Taylor Swift: Fearless	0.881
Taylor Swift: Hey Stephen	0.879

Figure 5 presents Artist2Vec on two of its principal components. It shows a variety of cluster shapes with many on the right showing very clear segmentation whilst the centre and middle exhibit less defined clustering. The most defined clusters comprise singers whose primary language isn't English or are associated with highly distinctive genres, such as *choir* music, which are unlikely to overlap with others.

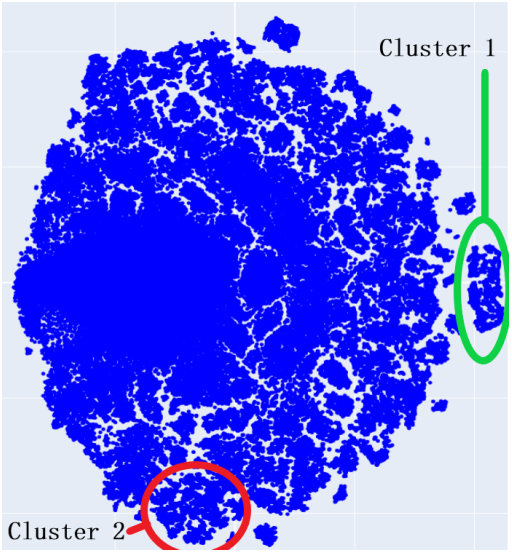

Figure 4. Visualisation of our Track2Vec embeddings. Cluster 1 holds Christmas songs whilst Cluster 2 holds a collection of Hip Hop and RNB songs.

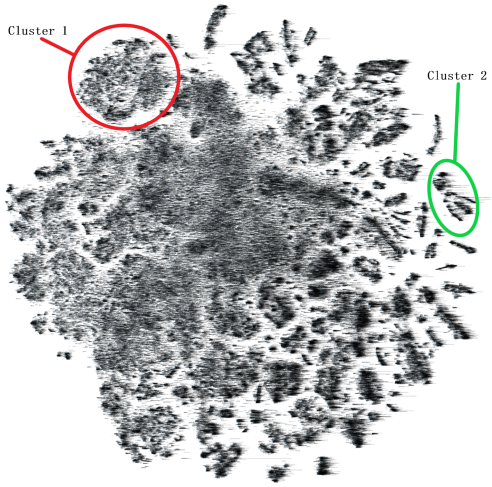

Figure 5. Visualisation of our Artist2Vec embeddings. Cluster 1 holds a collection of 2010's HipHop and RNB artists whilst Cluster 2 holds a variety of Indian music.

4.2. Experiment Setup

4.2.1 Metrics

The evaluation strategy precisely mirrors what is set out in the competition guidelines [1] - this aims to make our approach directly comparable with existing solutions. In each

of the following equations, R refers to an ordered list of recommended tracks, while G is the ground truth set of tracks.

Equation 4 is the R-precision which is the number of predicted relevant songs in the Top 500 divided by the number of known relevant songs.

$$\text{R-precision} = \frac{|G \cap R_{1:|G|}|}{|G|} \tag{4}$$

The second metric is the Normalized Discounted Cumulative Gain (NDCG, Equation 7) which is an order-aware metric of the 500 songs. This is calculated by calculating DCG and dividing it by the ideal DCG where all recommended tracks are perfectly ordered by their relevance.

$$\text{DCG} = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{\log_2 i} \tag{5}$$

$$\text{IDCG} = \sum_{i=2}^{|G \cap R|} \frac{1}{\log_2 i} \tag{6}$$

$$\text{NDCG} = \frac{DCG}{IDCG} \tag{7}$$

The final measure is 'clicks' which calculates the number of times the end-user would have to refresh the recommendations to obtain a relevant track.

$$\text{clicks} = \left\lfloor \frac{\arg\min_i \{R_i : R_i \in G|\} - 1}{10} \right\rfloor \tag{8}$$

The final leaderboard of submissions is obtained through a *Borda Count* rank aggregation over the R-precision, NDCG, and clicks metrics.

4.2.2 Playlist Loss

To train the deep learning models in Section 3.2 and 3.3, the evaluation metrics are incorporated into a custom loss function 'playlist_loss'.

$$\text{loss}_r(i) = 1 - \text{R-precision}(i) \tag{9}$$

$$\text{loss}_{\text{ndcg}}(i) = 1 - \text{ndcg}(i) \tag{10}$$

$$\text{loss}_{\text{clicks}}(i) = w_{\text{clicks}} \times \text{clicks}(i) - \frac{1}{50} \tag{11}$$

$$\text{loss}_i = w_r \times \text{loss}_r(i) + w_{\text{ndcg}} \times \text{loss}_{\text{ndcg}}(i) + \text{loss}_{\text{clicks}}(i) \tag{12}$$

4.2.3 Setup & Hyperparameter Tuning

A single NVIDIA 1080Ti GPU was used for training Track2Vec and Artist2Vec. Track2Vec took 5 hours to pre-train, and 2 hours for Artist2Vec. Hyperparameter tuning was conducted manually for the embedding models, the process involved adjusting the following variables `context_size`, `dimensions`, `epochs`, `min_track_freq`. Each setup was evaluated by interrogating the nearest songs on cosine distance to popular songs (Table 4). The final hyperparameters are stated in Section 2.

4.3. Results

The competition only permits 5 submissions per group, so to make the most of this allowance we decided to put forward the variations of the **Mean Pooling** and **Enriched Embeddings** models. Whilst the LSTM approach held promise, based on limited local evaluation using the loss function in Equation 12, the training was insufficient and failed to converge.

At the time of writing, we rank **58th on the global leaderboard out of 253 teams**. Despite the low R-Precision of 0.092, the NDCG score of 0.191 indicates that the top predicted tracks are more likely to be relevant. This is a very positive result given the simplicity of the mean-pooling approach.

Table 5. Official Results on 2018 ACM RecSys Challenge

	R-Precision	**NDCG**	**Clicks**
Mean Pooling	**0.092**	0.168	10.668
Mean Pooling (top-n)	0.083	0.190	8.338
Mean Pooling (sorted top-n)	0.088	**0.191**	**7.520**
Enriched Embeddings	0.073	0.118	24.3

5. Conclusion

Through utilizing techniques inspired by Natural Language Processing, this project contributed high-quality embedding representations for Spotify tracks, artists, and playlist titles. These embeddings form the basis of our approach to playlist continuation. We developed three models with increasing complexity: **(1)** Track2Vec Mean Pooling **(2)** Enriched Embeddings **(3)** LSTM embedding aggregation. The official result of 58th out of 252 teams on the 2018 ACM RecSys Challenge validates the embedding approach for playlist continuation and signposts for future development.

5.1. Future Work

Due to time constraints, the full training of the LSTM model for embedding aggregation has not been possible. The flexibility of this approach offers promise for future attempts at this challenge. The embedding quality could be improved by pre-training transformer-based embeddings over the dataset [9].

References

[1] *RecSys Challenge '18: Proceedings of the ACM Recommender Systems Challenge 2018*. New York, NY, USA: Association for Computing Machinery, 2018.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[3] M. Volkovs, H. Rai, Z. Cheng, G. Wu, Y. Lu, and S. Sanner, "Two-stage model for automatic playlist continuation at scale," in *Proceedings of the ACM Recommender Systems Challenge 2018*, ser. RecSys Challenge '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3267471.3267480

[4] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2021.

[5] H. Yang, Y. Jeong, M. Choi, and J. Lee, "Mmcf: Multimodal collaborative filtering for automatic playlist continuation," in *Proceedings of the ACM Recommender Systems Challenge 2018*, ser. RecSys Challenge '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3267471.3267482

[6] H. Zamani, M. Schedl, P. Lamere, and C.-W. Chen, "An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation," 2019.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[12] B. Vintch, "Quick lists: Enriched playlist embeddings for future playlist recommendation," *CoRR*, vol. abs/2006.12382, 2020. [Online]. Available: https://arxiv.org/abs/2006.12382

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735