Set Transformers for Playlist Recommendation

Matias Bronner, Brendan Rathier, Daniel Schiffman, Camilo Tamayo-Rousseau

CSCI1470

Defining the Problem

Creating cohesive music playlists is a core part of the modern listening experience. Feeling dissatisfied with the playlist recommendation systems on Spotify and Apple music we sought to explore how we could create a model to **learn the "vibe" of a playlist** to create a recommendation system. Specifically, **given a playlist title and songs with musical features,** our goal was to **predict 5 songs that are appropriate for the playlist.**

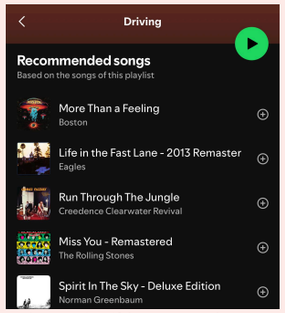


Figure 1. Spotify's recommendation system: an inspiration for our project.

Data

For the data for our project, we used a dataset of **1 million user playlists** from Spotify from 2010-2017 [1]. We then joined this dataset with 4 other datasets [2] [3] [4] [5] that included features about songs from the Spotify API such as genre, tempo, popularity, danceability, duration, etc. Therefore, we had user playlists with titles and a variety of descriptive features for each song in the playlist.

Architecture

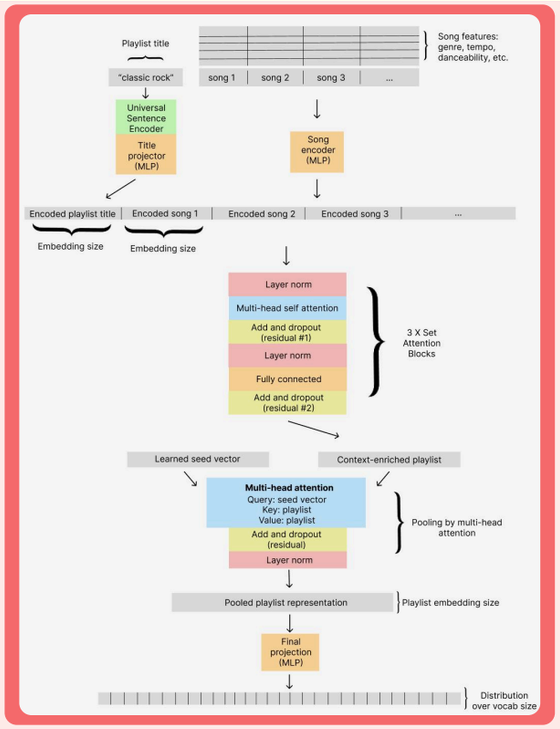


Figure 2. We use a pre-trained text encoder [6] for playlist titles, a **set transformer** with attention and **pooling by multi-head attention** [7] to make a final playlist representation from which we predict a distribution over the vocab size.

Training

Loss Function

We use the following custom **weighted approximate-rank pairwise (WARP)** [8] loss function: where $H > S$ and $w(k) = \sum_{r=1}^{k} \frac{1}{r}$ and margin = 1.

For every song in the original playlist P (positives):

Compute the positive score: $s_i = s(P, i)$

Sample negatives until 50 or a violation:

Compute negative score: $s_j = s(P, j)$

Violation if: $s_j > s_i - \text{margin}$

\# of negative samples we had to take = k

Compute the weighted WARP loss:

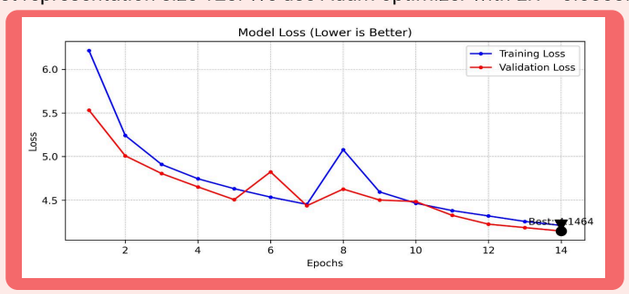
$L = H \cdot w(k) \cdot \max(0, 1 + s_j - s_i)$ if song was hidden

$L = S \cdot w(k) \cdot \max(0, 1 + s_j - s_i)$ if song was seen

Final loss for the playlist L_P is the average of the loss for all of the positives

Performance

We train our model for 14 epochs (so far) with seen weight S = 0.5, hidden weight H = 1.0, hiding between 0.5 and 0.1 of the playlists, song embedding size 128 and playlist representation size 128. We use Adam optimizer with LR = 0.0003.



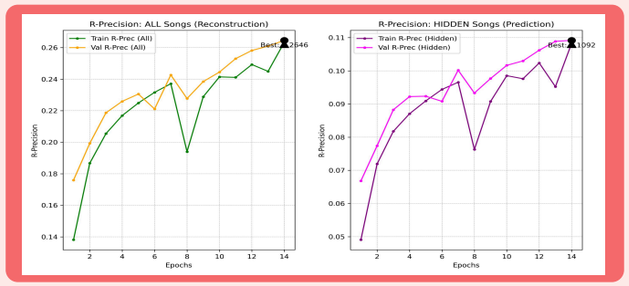


Figure 3. To assess our model's performance we use a metric called R-precision [6]. If a playlist has length R, R-precision is the proportion of songs in the top R predictions of the model that were in the original playlist. We track both a "hidden R-precision" and a "total R-precision" to examine how our model is performing at the reconstruction aspect of the task vs. the prediction aspect.

Ablation Testing

Figure 4. We run our model on approximately 40% of our dataset for 10 epochs also hiding between 0.5 and 0.1 of the playlists to assess the impact of our weighting hyperparameters in our loss function. We see inconclusive results from adjusting "seen" and "hidden" weights and we see apparently negative impacts on performance by increasing embedding dimension sizes.

TABLE I: Ablation Tests for "seen weight" and "hidden weight"

Seen Weight	Hidden Weight	(Best) Validation R-precision (Total)	(Best) Validation R-precision (Hidden)
0.5	1.0	0.224	0.093
0.25	1.0	0.215	0.086
0.10	1.0	0.217	0.089

TABLE II: Ablation Tests for "song embedding size" and "playlist embedding size"

Song Embedding Size	Playlist Embedding Size	(Best) Validation R-precision (Total)	(Best) Validation R-precision (Hidden)
128	128	0.224	0.093
128	256	0.223	0.089
256	128	0.207	0.083
256	256	0.103	0.034

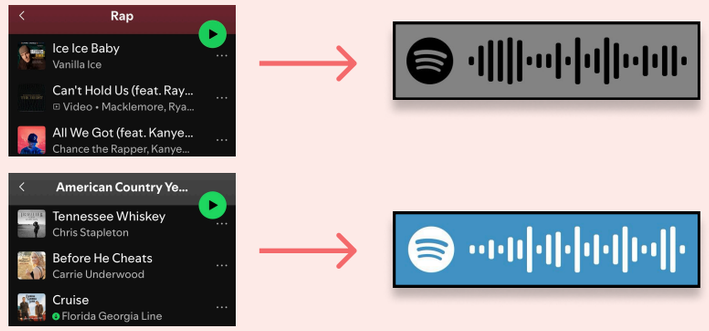


Figure 5. We see some strong recommendations for a "rap" playlist but we see mode collapse on other examples.

Future Work

Future directions could include using the text embeddings for song and artist names or audio data in our model or trying to reduce mode collapse by utilizing regularization techniques to encourage less confident logits.

Acknowledgements

We would love to thank our TA mentor Armaan for his feedback and support and Professor Ewing for a great semester!

References

[1] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani, "Recsys challenge 2018: Automatic music playlist continuation," in Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 527–528.; [2] https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs; [3] https://www.kaggle.com/datasets/amitanshjoshi/spotify-1million-tracks; [4] https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?select=tracks.csv; [5] https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset/data; [6] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," CoRR, vol. abs/1803.11175, 2018. [Online]. Available: http://arxiv.org/abs/1803.11175; [7] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in International conference on machine learning. PMLR, 2019, pp. 3744–3753; [8] A. Ferraro, D. Bogdanov, J. Yoon, K. Kim, and X. Serra, "Automatic playlist continuation using a hybrid recommender system combining features from text and audio," in Proceedings of the ACM Recommender Systems Challenge 2018, 2018, pp. 1–5;