Music Recommendation System

P17: Srivathsan Govindarajan, Naren Sivakumar Rajalakshmi,
Sri Roopa Ramesh Babu, Prasanna Venkatesh Thambuanantharaman
Department of Computer Science, North Carolina State University
{sgovind7, nsivaku6, srames25, pthambu}@ncsu.edu

1 Background & Introduction

Music streaming revenues increased to 17.5 billion dollars a year in 2022, registering over a six fold increase from 2015. With such explosive growth there is a constant need for improvements in the user experience, as that is one of the key areas where streaming giants can differentiate themselves. A core aspect of the user experience is the recommendation system which recommends new songs to listeners. One of the reasons behind Spotify's meteoric rise has been their excellent recommendation system, showing us that users respond positively to a great user experience. The goal of our project is to build a recommendation system using transformers which leverages session details and the skip attribute found in Spotify's Sequential Skip Prediction Dataset. We leverage the skip attributes as the source of ground truth values to evaluate the recommendations.

Gabriel de Souza et al.[1] developed Transformers4Rec, an open-source library built upon Hugging-Face's Transformers library. Casper Hansen et al.[2] came up with CoSeRNN, a RNN based neural network architecture that also models Contextual and Sequential User Embeddings to recommend songs. Meggetto et al.[3] propose a technique for illustrating the importance of session-based skipping behavior in song recommendation engines, using data transformation and clustering algorithms on Spotify's Music streaming session dataset. Pavan et al.[4] employed the BERT4Rec transformer architecture to develop a session-based system that recommends songs to the user.

One common aspect of all these papers is that all of them implement a single input modality for all the models they have compared, while we want to build and train models with different input modalities.

2 Methodology

Our proposed plan is to build 3 models, each using a different approach and compare their performances in order to evaluate the best one. For our baseline model we use LGBM [6].

2.1 LightGBM

LightGBM is a gradient boosting algorithm that uses leaf oriented tree structures to establish a hierarchical relationship between considered attributes and their respective data points.This model considers data based on information gain and the gradient loss function which is brought about by the Gradient-based One-Side Sampling which ensures that only the data with high gradient value are considered for analysis and the Exclusive Feature Bundling part ensures that the sparseness of features are minimized and brought together. LightGBM was chosen as our baseline model basing its tendencies of handling categorical data and applying leaf level ensembling of decision trees in the learning process.

2.2 Behaviour Sequence Transformer

Our second model, and the main focus of this project, would be to use a Transformer [5] whose architecture is shown in Fig 1. A Transformer model is a neural network that applies mathematical techniques called attention or self-attention to learn context and the relationships between data elements. Transformers make use of an encoder and decoder to estimate the probability of what the next data element should be in the output. In our case, we implement a specialized type of Transformer known as Behaviour Sequence Transformer which is designed to capture sequential patterns present in the user's behaviour sequences for recommendation. We also employ a embedding lookup layer which contains the track metadata features for the transformer to refer to during training.

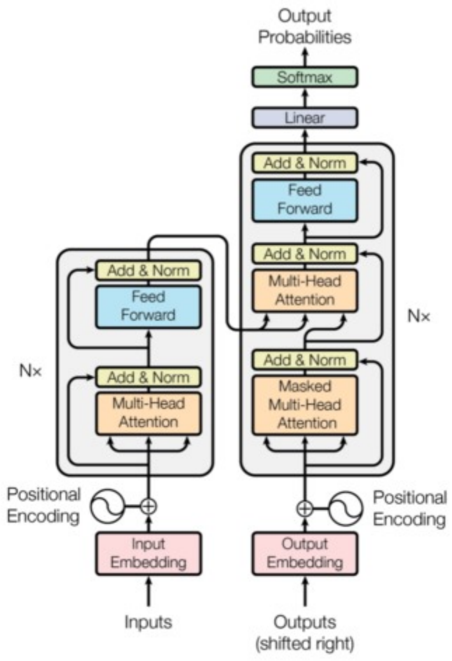

Figure 1: Transformer Architecture

2.3 LightGCN

The third and final model in our analysis is LightGCN [7], which is a graph-based recommendation model derived as a simplified version of Neural Graph Collaborative Filtering (NGCF) [8] and leverages the property of neighborhood aggregation of nodes from the given graph. The architecture of LightGCN is shown in Fig 2.

In Light Graph Convolution, the normalized sum of neighbor embeddings is propagated to the next layer and in the aggregation step we sum over the embeddings at each layer to obtain the final representations. This Light graph convolution operation is represented by the equation:

$$\begin{aligned} \mathbf{e}_u^{(k+1)} &= \textstyle\sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)} \\ \mathbf{e}_i^{(k+1)} &= \textstyle\sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)} \end{aligned}$$

Where e_u^k and e_i^k are user and item embeddings at kth layer respectively. After K layers LGC, we further combine the embeddings obtained at each layer to form the final representation of a user (an item):

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}$$

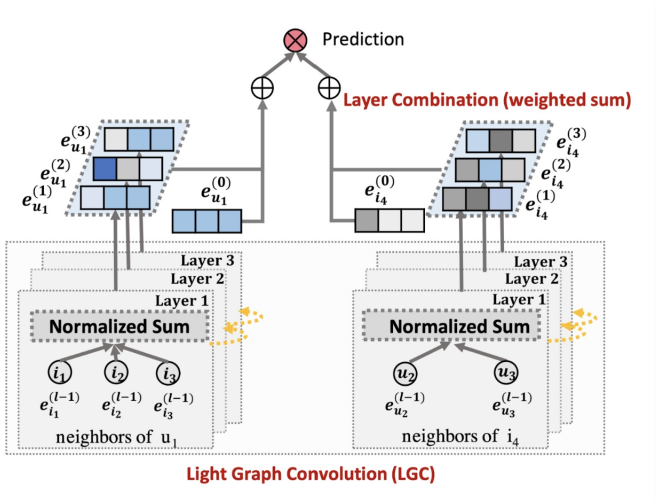

Figure 2: LightGCN Architecture

where $\alpha_k \geq 0$ is a hyperparameter denoting the importance of the k-th layer embedding in constituting the final embedding. The final prediction is defined as the inner product of user and item final representations which is treated as our recommender score:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i$$

3 Experimental Setup

3.1 Dataset

We are using the Spotify Sequential Skip Prediction Dataset (mini version). The sessions in it contain at most 20 tracks. The dataset contains two data files, which are session information containing $\sim$ 170k rows and 21 attributes, and track metadata containing $\sim$ 40k rows and 20 attributes. We utilize the skip_1 attribute as our class label which indicates if the user skipped the track in it's initial stages .

3.2 Hypotheses

Our initial hypothesis, and the motivation behind choosing a Transformer as one of our models, is related to the attention mechanism. We hypothesize that the Transformer's attention mechanism, which helps it learn context and relationships between data elements, would make the model perform better. By implementing a Behavior Sequence Transformer we would further improve the model's performance as it captures sequential patterns and thus would be ideal for our use case where we have to capture patterns in sessions.

We also had the following question: Will there be a significant difference in the performance of our recommendation systems if the input data structures that are given to our models are different? Hence we try out a tabular data structure (using LightGBM), a sequential data structure (using Transformer) and a graph data structure (using LightGCN).

3.3 Exploratory Data Analysis

To understand the data set, we started by analyzing the pause trends of each track on a particular session against the user skip behavior which can be seen in Fig 3. All three *pause* attributes did not have any influence on track skipping. To investigate the impact of the time of day on user-track skipping behavior, we created visualizations that show its relationship with the skipping rate of each session. However, it's apparent that the time of day doesn't significantly influence whether a track is skipped (Fig 5). However, the action of the user skipping the previous track or not largely influences the current track's outcome as seen in Fig 6. We also observed a robust correlation between the

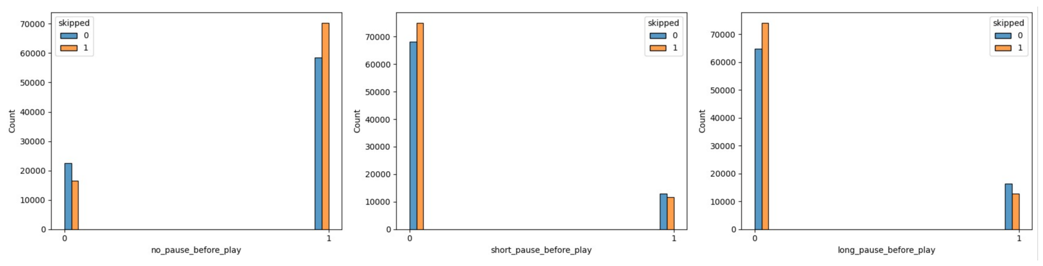

Figure 3: Histogram of pausing behavior with skipping variations.

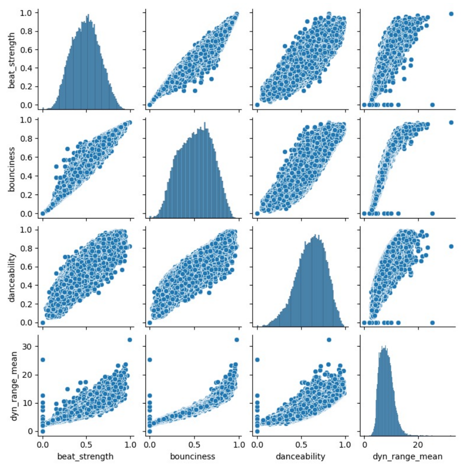

Figure 4: Pairplot displaying the relationship between the attributes dynamic range, danceability, bounciness, and beat strength present in the dataset.

variables 'beat strength,' 'bounciness,' 'danceability,' and 'dyn range mean,' supporting the intuitive notion that a song's beat intensity enhances its danceability and dynamic characteristics (Fig 4).

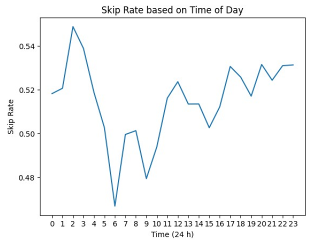

Figure 5: Skip rate based on time of the day the track was played.

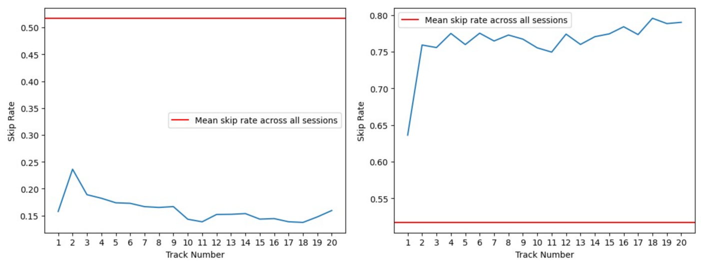

Figure 6: Skip rate of current track based on whether the previous track was skipped(right) or not skipped(left).

3.4 Preprocessing

Each of the model used in this paper requires a different input modality so a separate preprocessing stratergy is carried out. Hence the train/val/test (80/10/10) split is done beforehand to ensure that the models are comparable.

3.4.1 LightGBM preprocessing

LightGBM algorithm supports both discrete and Categorical data which will support the analysis of various categorical attributes of the data which are crucial for inferencing. Information on previously skipped/non-skipped tracks is also required for session-based analysis, this is done by concatenating the data points by grouping the entries based on session ID and shifting the data point by one-unit intervals. This maps the current data with the previously played track and its corresponding acoustic and behavioral features in that session. Other aspects of preprocessing include encoding Categorical data that will be bundled efficiently by the LGBM algorithm.

3.4.2 Transformer preprocessing

Transformers require their data to be in sequences in units of tokens. To mimic our dataset in this format, we consider each session as a sequence containing the list of tracks played, their corresponding acoustic features, historical skip behavior, and session behavior. The sequences are generated by aggregating these features across each session with a sequence length of 10 and step size of 3. These values are chosen with the intention that the minimum session length is 10 and we would want to predict the next top 3 tracks to be recommended.

3.4.3 LightGCN preprocessing

Since sessions and tracks are linked by interactions between them, we can treat them as nodes of a bipartite graph, and the edges represent if a track was played in that session. From the session-track interaction matrix $\mathbf{R}$ we obtain the adjacency matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix}$$

The above matrix is used as input to LightGCN to generate the embeddings for session and track nodes which is passed as signal to further layers to train the model through neighbourhood aggregation.

3.5 Evaluation Criteria

We plan to use ROC and Recall of top k recommended tracks as our metrics, by predicting whether a user skips the recommended track. In the case of LightGBM only the next track prediction is obtained so we only use the top track. We have decided to utilize the recall metric because the cost of false negatives (classifying a song that the user likes as not liked) is greater than false positives for recommendation systems.

3.6 Hyperparameters

To enhance our models' performance, we tuned the hyperparameters of our models over a defined search space. After extensive experimentation we have found the following hyperparameters (Table 1) to provide the best results.To prevent the model from getting overfitted on the data, Early Stopping was used with a patience value of 3. We also experimented with two different negative edge sampling stratergies namely, random negative and hard negative sampling. Fig 7 shows the training and validation loss curves and their convergence observed while fitting the BST and LightGCN models. We also want to highlight the Bayesian Personalized Ranking loss used for training the LightGCN model which tries to increase the difference between the positive and negative recommendations.

$$\mathbf{BPRLoss(i)} = \frac{1}{|\eta(\mathbf{i})|} \sum_{(i,j+)\epsilon\eta(i)} \mathbf{log}\sigma(\mathbf{score(i,j+)} - \mathbf{score(i,j-)})$$

where $(\mathbf{i},\mathbf{j}+)$ is a positive edge and $(\mathbf{i},\mathbf{j}-)$ is a negative edge.

Table 1: Hyperparameters

Model	Hyperparameter	Value
LightGBM	#Leaves	31
	Learning Rate	0.05
	#Estimators	100
	Loss	Binary Log
BST	#Layers	1
	Learning Rate	0.01
	#Hidden Units	128
	Optimizer	Adagrad
	Loss	BCE
LightGCN	#Layers	2
	Embedding Size	64
	Learning Rate	0.01
	Optimizer	Adam
	Loss	Bayesian Personalized Ranking

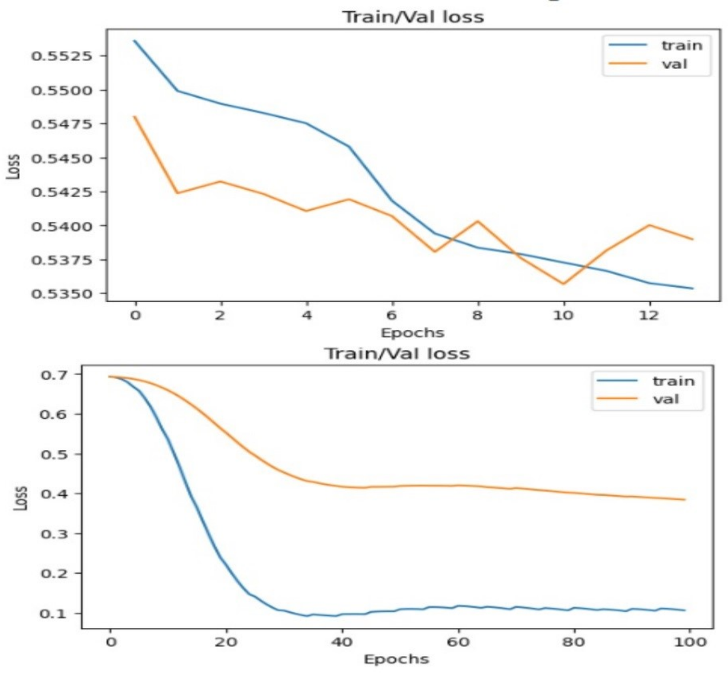

Figure 7: Train/Val loss of BST (above) and LightGCN (below)

4 Results

After fitting and tuning our models, we assessed them with recall and AUC metrics. From the above graphs, we can see that the BST model outperforms the baseline model (i.e., the LightGBM model) with by 7% in AUC and 4% in recall. In the overall picture, the LightGCN model trained using hard negative sampling emerged as the top performer with an AUC score of 97.70 and a recall of 93.40.

As in our hypotheses, we focused on viewing our problem statement from various perspectives, i.e., giving the input data to the models in different data structures. Based on the assessment, we can evidently see that the structure of the data that is dealt with for training our models makes a big difference. As the LightGCN model visualizes the data as a graph-like structure, it helps the model

Table 2: Results of the models

Model	AUC	Recall
LightGBM	73.10	71.40
BST	80.10	76.90
LightGCN (hard)	**97.70**	**93.40**

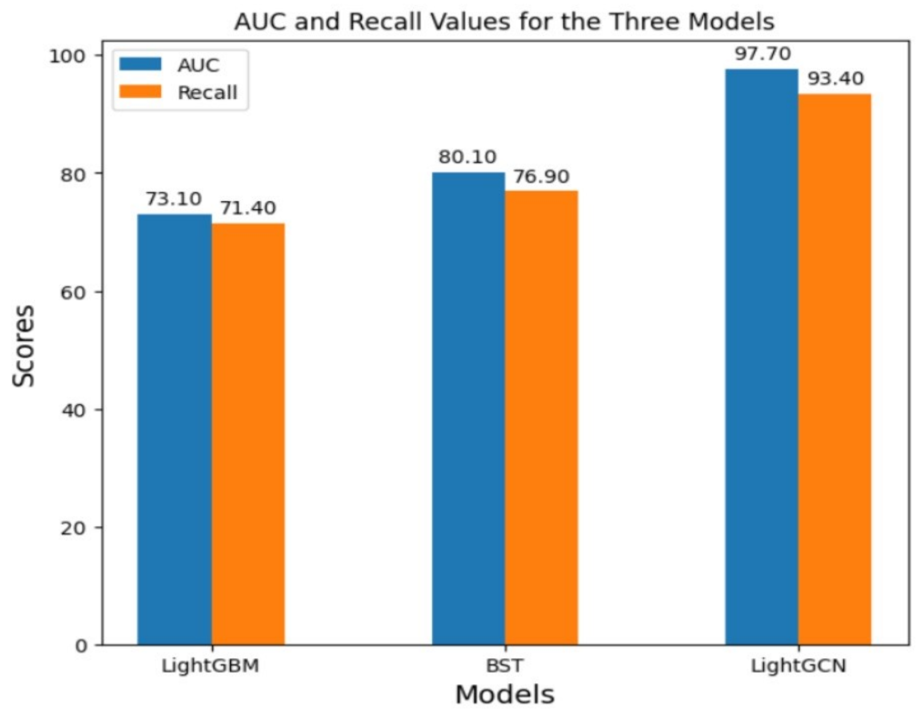

Figure 8: Results of the models

understand the relationship between each session and the related tracks, as it views the relationships in the data globally. Therefore the model only recommends songs closely related to the session which explains the high AUC and Recall scores obtained. .

5 Conclusion

Through this project, we understood that the performance of a model can be greatly impacted by how the data is structured. Our findings show that the use of graphs to visualize the relationship between the session and the tracks related to it can greatly improve the recommendation engine's quality. Our future work would involve applying the same process on the entire Spotify dataset instead of implementing it on a subset of the data. We also plan to leverage the use of a graph-based transformer and observe its performance.

6 References

[1] de Souza Pereira Moreira, G., Rabhi, S., Lee, J. M., Ak, R., Oldridge, E. (2021, September). Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. *15th ACM Conference on Recommender Systems* (pp. 143-153).

[2] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20). Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/3383313.3412248

[3] Francesco Meggetto, Crawford Revie, John Levine, and Yashar Moshfeghi. 2021. On Skipping Behaviour Types in Music Streaming Sessions. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21). Association for Computing Machinery, New York, NY, USA, 3333–3337. https://doi.org/10.1145/3459637.3482123

[4] Seshadri, P. and Knees, P., 2023. Leveraging Negative Signals with Self-Attention for Sequential Music Recommendation. arXiv preprint arXiv:2309.11623.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[6] Ke, Guolin et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." Neural Information Processing Systems (2017).

[7] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y.,& Wang, M. (2020). LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. ArXiv. /abs/2002.02126

[8] Wang, X., He, X., Wang, M., Feng, F., & Chua, T. (2019). Neural Graph Collaborative Filtering. ArXiv. https://doi.org/10.1145/3331184.3331267

[9] Brost, B., Mehrotra, R., and Jehan, T. The music streaming sessions dataset. In Proceedings of the 2019 Web Conference. ACM, 2019.

Code Repository

Github Repository

7 Appendix

7.1 Data Collection and cleaning

- Sri Roopa Ramesh Babu, Srivathsan Govindarajan, Naren Sivakumar Rajalakshmi, Prasanna Venkatesh Thambuanantharaman

7.2 Exploration

- Pause and correlation analysis: Sri Roopa Ramesh Babu

-Skip Attribute Analysis: Srivathsan Govindarajan

-Encoding input features into sequences: Prasanna Venkatesh Thambuanantharaman

7.3 Preprocessing and Model Development

- LightGBM: Sri Roopa Ramesh Babu

- Transformers: Naren Sivakumar Rajalakshmi, Prasanna Venkatesh Thambuanantharaman

-LightGCN: Srivathsan Govindarajan

7.4 Results Analysis

- Sri Roopa Ramesh Babu, Srivathsan Govindarajan, Naren Sivakumar Rajalakshmi, Prasanna Venkatesh Thambuanantharaman

7.5 Conclusion Drawing

- Sri Roopa Ramesh Babu, Srivathsan Govindarajan, Naren Sivakumar Rajalakshmi, Prasanna Venkatesh Thambuanantharaman

7.6 Presentation Preparation

- Sri Roopa Ramesh Babu, Srivathsan Govindarajan, Naren Sivakumar Rajalakshmi, Prasanna Venkatesh Thambuanantharaman

7.7 Final Report Creation

- Sri Roopa Ramesh Babu, Srivathsan Govindarajan, Naren Sivakumar Rajalakshmi, Prasanna Venkatesh Thambuanantharaman