Exploring Transformers in Embedding-Based Music Recommendation

Vinicius Sylvestre Simm, Douglas Rorie Tanno and Marcos Aurelio Domingues
Department of Informatics
State University of Maringá
Maringá, Paraná, Brazil
{pg404811, pg404806, madomingues}@uem.br

***Abstract*—Music recommendation systems help users explore large music collections by suggesting tracks they might enjoy. This work focuses on session-based music recommendations, where suggestions are made using only the sequence of songs played during a single listening session, without any prior user history. We compare three approaches for generating recommendations based on vector representations (embeddings) of songs. The first is a simple model that averages the embeddings of all tracks in a session. The second one uses an LSTM neural network. The third approach uses a Transformer neural network, which is better designed to understand the order and context of the songs. In all cases, the models generate a vector to compute the cosine similarity to all known item embeddings and to recommend the most similar tracks. These embeddings are created using the Word2Vec algorithm, with both Skip-Gram and CBOW variations. The results show that the Transformer model achieves better recommendation quality than the baseline, despite requiring more computational resources.**

***Index Terms*—Recommendation Systems, Embeddings, Session-Based Recommendation Systems, Neural Networks, LSTM, Transformers.**

I. Introduction

In recent years, access to digital content, including music, movies, and books, has expanded dramatically, driven by the widespread availability of the Internet and the rapid growth of streaming platforms. As users are exposed to an ever-increasing amount of content, recommendation systems have become essential tools to help them navigate through vast catalogs and efficiently discover relevant items.

Recommendation systems help users navigate the vast amount of content on digital platforms. In the music domain, this task is more complex due to the diversity of genres and the influence of social and geographic factors on preferences. Many systems rely on user data and ratings to detect patterns and generate suggestions [1].

Music preferences are often context-dependent. A user may prefer different songs depending on his/her context, and these short-term preferences can be inferred from the sequence of recently played tracks [2]. To capture this behavior, user activity is typically grouped into sessions, which are sequences of interactions without long breaks.

Depending on the availability of user identification, we can define two types of recommendation tasks: session-based and session-aware. Session-based models use only the current session, which is useful in anonymous or cold-start scenarios [3]. On the other hand, session-aware models use data from previous sessions to build more personalized recommendations [4].

Recommendation systems often deal with items such as songs, movies, or books, which are usually represented by simple identifiers in the dataset. Although these identifiers are useful for referencing items, they do not carry information about the content or similarity among them. To generate more meaningful and accurate recommendations, it is necessary to convert these items into richer representations. A widely used solution is to employ embeddings, vector representations that map each item into a low-dimensional space. In this space, items that are similar in terms of user behavior or context tend to appear closer together. The embeddings are typically learned from user interaction patterns, such as sequences of items consumed within sessions, allowing the system to capture latent relationships among them.

Recent advances in sequence modeling have highlighted the effectiveness of Transformer-based architectures in capturing temporal and contextual relationships. Initially introduced for natural language processing tasks [7], Transformers have since been adapted to various domains, including recommendation systems, due to their ability to model dependencies among items without relying on recurrence.

In this paper, we investigate the effectiveness of embedding-based recommendation models in a session-based music recommendation scenario. We compare three distinct approaches: a baseline model, which generates recommendations by averaging the embedding vectors of tracks in a session and retrieving the closest items based on cosine similarity; a sequential model, which uses an LSTM neural network architecture; and a Transformer-based model, which uses a deep neural network architecture designed to capture sequential patterns within sessions and predict the next item in the form of an embedding vector. All approaches rely on pre-trained item embeddings generated through Word2Vec with both Skip-Gram and CBOW variants.

To assess the models' performance, we use various evaluation metrics, including Hit Rate, Precision, Recall, F-score, MRR, NDCG, MAP, Coverage, and Popularity Bias. The experiments were conducted on two real-world music listening

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

datasets. The results show that Transformer-based models consistently outperform the baseline and the LSTM model regarding recommendation accuracy, highlighting their ability to understand session dynamics better. However, a trade-off was observed with reduced catalog coverage. These findings suggest that Transformer architectures are a promising direction for session-based music recommendation, especially when sufficient computational resources are available.

The remainder of this paper is structured as follows. Section II reviews related work. Section III details the methodology, covering datasets, data preprocessing, training and test set splitting, recommendation models, and evaluation setup. Section IV presents and discusses the experimental results. Finally, Section V concludes the paper with final remarks and future research directions.

II. Related Work

Recommendation systems have been developed to help users navigate large item catalogs, even in situations where they have little or no prior experience with the available items [8]. One of the key challenges in this area is finding a balance between satisfying users with personalized recommendations and achieving business goals such as increasing sales or enhancing customer retention. At the same time, these systems also raise concerns about reinforcing unhealthy consumption patterns and behaviors [9].

Such systems are widely used in e-commerce platforms to recommend products such as books and clothing, and in streaming services to suggest music, movies, and TV shows. However, music recommendation differs from other domains in several ways: songs are much shorter than items like movies or travel packages; the number of items available is extremely large; inaccurate recommendations tend to have less negative impact; content-based information, such as audio features, can be extracted from songs; repeated recommendations are often welcome; and recommendations are usually consumed sequentially and frequently passively [10].

The recommendation systems can also differ significantly in how they handle user interaction history. Session-aware systems use information from both the current session and previous sessions associated with the user. This enables them to create long-term user profiles and generate more personalized recommendations across various contexts. However, this approach requires persistent user identification and storage of historical data, which may not always be available or desirable [4].

In contrast, session-based systems rely solely on the interactions within the current session, without using any information from previous sessions [3], [11]. This makes them particularly suitable for scenarios where users are anonymous, not logged in, or using the system for the first time. Despite the limited input, session-based methods aim to capture short-term intent by analyzing the sequence of recent actions.

Given the practical advantages and growing research interest in session-based recommendation, particularly in privacy-sensitive or cold-start environments, this approach is the one that we explore in our work.

To enable recommendation models to better understand and compare items, it is common to represent each item as a dense vector in a low-dimensional continuous space, called an embedding. The embeddings are constructed so that items with similar usage patterns, such as songs frequently played in the same session, are close together in the vector space. This approach allows the model to capture latent semantic relationships among items based on their co-occurrence in user behavior data, rather than relying on explicit metadata or user profiles. The concept of embeddings has its origins in natural language processing, notably with the introduction of Word2Vec by [12], which showed how words that appear in similar contexts tend to have similar vector representations. This idea has since been extended to recommendation systems, where items are treated analogously to words, and sessions or user interactions are treated analogously to sentences or documents [13].

A widely used method for generating these embeddings is Word2Vec, which was initially developed for natural language processing tasks [12]. In this method, sequences of items are used as input to a model that learns representations based on their context within sessions. Word2Vec has two main architectures: the Skip-Gram, which predicts surrounding items given a target, and the Continuous Bag-of-Words (CBOW), which predicts the target item from its context. Both approaches aim to place items that frequently appear together in close positions in the embedding space, making them useful for vector similarity-based recommendations.

Long Short-Term Memory (LSTM) networks [5] enhance traditional neural architectures by introducing gating mechanisms that control the flow of information across time steps. These gates enable the model to selectively retain or discard information, making LSTM networks particularly well-suited for modeling sequential data, such as user sessions. Prior research has successfully applied LSTM models in recommendation systems to capture temporal dependencies in user interaction sequences [6]. By incorporating the sequence of interactions as an additional source of information, these models can learn short-term item transitions and behavioral patterns, ultimately improving recommendation accuracy.

More recently, Transformer-based models have gained significant attention in the field of recommendation systems due to their ability to model complex dependencies and contextual relationships within sequences. Transformers leverage self-attention mechanisms to capture interactions among all elements in a session, regardless of their position. This makes them particularly effective in session-based recommendation tasks, where the sequence and co-occurrence of items contain rich behavioral information [7], [14].

However, many Transformer-based recommendation methods rely on the joint training of item embeddings and the sequence model, which can obscure the individual contributions of each component and reduce reusability across models or domains. Furthermore, these models often require access

to large-scale datasets and high computational capacity, which limits their accessibility and reproducibility.

In contrast, this work proposes to assess the capacity of a Transformer model to generate effective recommendations using pre-trained and fixed embeddings. This decoupled approach offers greater modularity and interpretability, allowing for more focused experimentation on how well a Transformer can leverage sequential context when embeddings are held constant. A simple baseline using the average of item embeddings and cosine similarity provides a point of comparison to measure the effectiveness of the LSTM and the Transformer's sequential modeling capabilities.

III. Methodology

This section describes the steps taken to create and evaluate the proposed session-based recommendation models. The process starts with the preparation and filtering of user interaction data, followed by session anonymization and segmentation. After isolating the test data, we generate item embeddings using the Word2Vec algorithm in both Skip-Gram and CBOW variants. These embeddings are then used in three recommendation strategies: a baseline model based on the average of vectors within a session, a second model that uses an LSTM architecture, and a third model made with a Transformer architecture to predict the next item. Each of these steps is described in detail in the following sections. The whole process is illustrated in Figure 1.

A. Data Preprocessing

For this work, we used two publicly available music listening history datasets: Xiami [2] and Music4All [15]. Both datasets contain user interactions with songs, along with metadata such as timestamps and, in one of the datasets, audio characteristics. For the purposes of this work, however, we focused exclusively on the sequential playback data, ignoring any additional content or user-level information.

To construct sessions, interactions were grouped by user and split whenever there was a gap of at least 30 minutes between consecutive plays. Each resulting session represents a continuous listening experience. After segmentation, the data was anonymized by removing all user identifiers to simulate a realistic session-based scenario without user tracking.

Short sessions with three or fewer songs were discarded, as they would not provide sufficient context for learning short-term preferences. Then, for each session, we withheld the last song as a test item. To ensure proper evaluation, we kept only those sessions where the test song also appeared in the training set, ensuring that all items to be predicted had corresponding embeddings.

Throughout the process, the test set remained completely isolated and was not used to generate embeddings or train the model.

After preprocessing, the Xiami dataset comprises almost 180,000 sessions with an average length of 23 songs, while the Music4All dataset achieves more than 400,000 unique sessions, with an average of 11 songs per session.

B. Embedding Generation

To represent each song in a session as a dense vector, we trained item embeddings using the Word2Vec algorithm, originally developed for word representation in natural language processing [12]. In our context, each session was treated as a sentence, and each song was treated as a word, preserving the sequential nature of the interactions.

Both Skip-Gram and CBOW variants of Word2Vec were explored in our work. To improve the quality of the embeddings, we applied a hyperparameter optimization process using the Optuna framework [16]. Optuna is a software library for automatic hyperparameter search with an efficient implementation of Bayesian optimization. It allows the definition of search spaces for each parameter and uses a sampling strategy to explore them, guided by a performance metric. The following hyperparameters have been tuned during the optimization:

- `vector_size`: Dimensionality of the embedding vectors. Values tested: $\{128, 256, 512, 1024\}$.
- `window`: Size of the context window for surrounding items. Range: 3 to 15.
- `alpha`: Initial learning rate. Logarithmic range: 10^{-4} to 10^{-1}.
- `min_alpha`: Minimum learning rate after decay during training. Logarithmic range: 10^{-4} to 10^{-2}.
- `negative`: Number of negative samples per positive sample. Range: 5 to 20.
- `sample`: Threshold for subsampling frequent items. Logarithmic range: 0.0001 to 0.01.
- `hs`: Binary choice between using hierarchical softmax (1) or negative sampling (0).

C. Recommendation Models

We explored three different models for generating recommendations, each based on the item embeddings obtained in the previous step.

The first model, used as a baseline and during the tuning of the Word2Vec embeddings, computes the mean vector of the item embeddings in a session. This averaged vector serves as a representation of the session. To generate recommendations, it is compared to all known embeddings using cosine similarity, and the top-10 most similar items are selected as recommendations.

The second model employs a Long Short-Term Memory (LSTM) neural network to learn sequential patterns in user sessions [5]. It operates directly on a sequence of pre-computed embeddings, thereby preserving the order of item interactions within a session. The network consists of a single LSTM layer with a dropout layer with a rate of 0.1 to reduce overfitting. A final dense layer projects the output back into the original embedding space.

The third model adapts a Transformer neural network designed to capture sequential dependencies within a session [7]. Instead of taking item indices as input, the model receives a sequence of embeddings, preserving the order of item interactions. The output is a single vector in the same space as the embeddings, representing the predicted next item

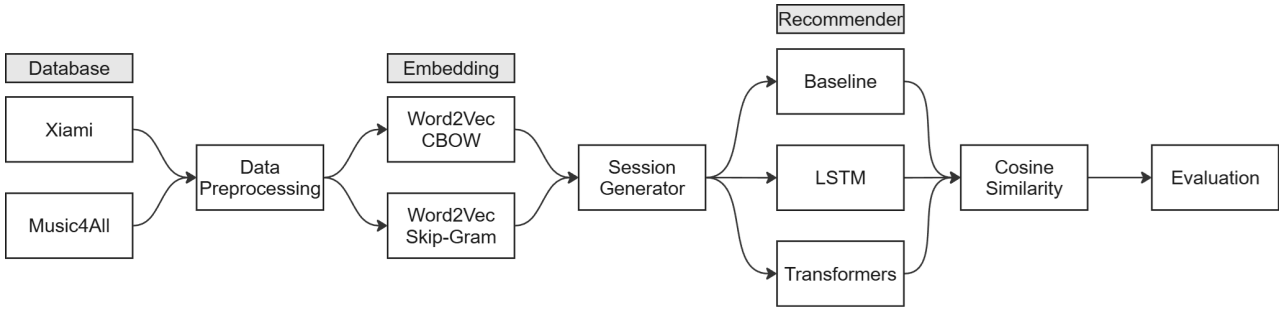

Fig. 1. Diagram with all major steps of our proposal.

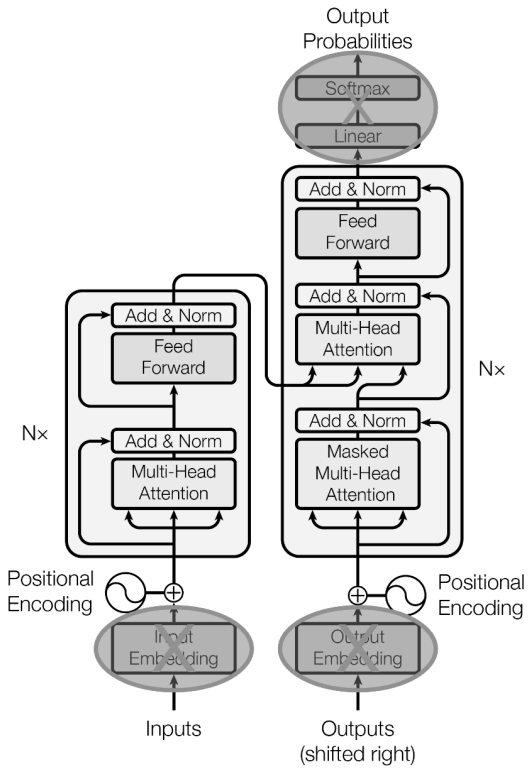

Fig. 2. Adaptation of the original Transformer model [7].

in the session. The Figure 2 demonstrates how the original architecture of the Transformer is adapted for this new task by excluding the embedding layer, as it is already provided in the input. The figure presents the original Transformer architecture with a significant modification. In this adapted version of the model, the original embedding layer becomes unnecessary, as both the inputs and outputs are expected to be precomputed embeddings. Therefore, we removed the layers marked in red in the image from the original Transformer model.

Similar to the baseline, recommendations are generated by computing the cosine similarity between the output vector and all item embeddings, returning the 10 songs with the highest similarity.

The main difference between the models lies in how the session representation is generated, either through simple averaging or by utilizing a learnable architecture that can model sequence patterns.

To efficiently compute the cosine similarity between the predicted vector and all known item embeddings, we used the FAISS library [17]. FAISS (Facebook AI Similarity Search) is a library developed by Facebook AI Research for fast nearest neighbor search in high-dimensional spaces. It provides optimized implementations that leverage multi-threading and GPU acceleration, making it particularly well-suited for large-scale recommendation tasks where speed is crucial. In our work, FAISS is used to index the item embeddings and retrieve the top-10 most similar items for each session vector.

D. Evaluation Setup

To assess the performance of the recommendation models, we adopted a session-based prediction setup. As illustrated in Figure 3, the last track of each session is withheld and used as the test target. All white blocks in the figure represent these held-out tracks, which are excluded from the training process and not used to generate embeddings or tune the recommendation system. During evaluation, all tracks preceding the final one serve as input to the model, which generates a ranked list of 10 recommended items. The withheld track serves as the ground truth and is used solely for evaluation purposes. Then, metrics are computed based on whether the true next item appears in the top-10 list and, if so, its position within that ranking.

To evaluate the performance of the recommendation models, we computed several evaluation metrics commonly used in the literature on recommendation systems. All metrics were computed based on the top-10 recommended items for each test instance.

- **Hit Rate@10 (HR@10)**: Measures the proportion of sessions in which the ground truth item appears among the top-10 recommended items. It reflects the model's ability to retrieve at least one relevant item [18].
- **Precision@10**: Indicates the fraction of recommended items in the top-10 that are actually relevant. Since each session has only one relevant item, the precision is either 0 or 0.1 [19].
- **Recall@10**: Measures the proportion of relevant items that were successfully retrieved. With only one relevant

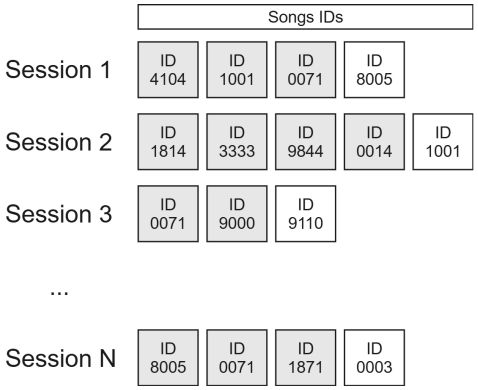

Fig. 3. Example of how the sessions are divided.

item per test case, recall is 1 if the item appears in the top-10 or 0 otherwise [19].

- **F-Score@10**: Harmonic mean of precision and recall. It provides a balanced view between retrieving relevant items and minimizing false positives.
- **Mean Reciprocal Rank@10 (MRR@10)**: Calculates the average reciprocal rank of the first relevant item in the recommendation list. If the correct item appears at position k, the reciprocal rank is $1/k$ [20].
- **Normalized Discounted Cumulative Gain@10 (NDCG@10)**: A ranking-sensitive metric that gives more weight to relevant items appearing earlier in the recommendation list. It captures the quality of the ranking order [21].
- **Mean Average Precision@10 (MAP@10)**: Computes the average of precision values at the rank position of the relevant item across all sessions. It evaluates both relevance and ranking [21].
- **Coverage**: Measures the proportion of unique items from the catalog that were recommended. Higher values indicate greater recommendation diversity [22].
- **Popularity Bias**: Calculates the average popularity of the recommended items. A higher score suggests that the model tends to favor frequently occurring (popular) items, potentially reducing personalization [23].

E. Training Procedure

Due to the large amount of data, the presence of long sessions, and the high dimensionality of the embeddings, memory management became a key challenge during the training process. To address these issues, a custom data generator was implemented to convert indexed session data into corresponding embedding sequences in real-time. This generator was responsible for creating mini-batches and feeding them directly into the GPU during training, thereby significantly reducing memory overhead and enabling the use of more complex model architectures.

By avoiding the need to store the entire embedding sequences in memory at once, this strategy enabled more efficient resource utilization and made it possible to experiment with deeper Transformer architectures and larger batch sizes. The generator ensured a continuous flow of preprocessed data without overloading the GPU memory, which was critical for sustaining the training process over large datasets.

All experiments were conducted on a machine equipped with an AMD Ryzen 9 9900X processor, 64 GB of RAM, and an NVIDIA RTX 4070 Super GPU with 12 GB of VRAM. This configuration was crucial for supporting the computational demands of both embedding training and Transformer model training.

IV. Results and Discussion

In this section, we present the results obtained with our empirical evaluation.

Initially, it was necessary to generate the item embeddings that would serve as the basis for all subsequent steps. Separate sets of embeddings were created for each combination of dataset (Xiami and Music4All) and Word2Vec architecture (Skip-Gram and CBOW). To ensure the appropriate selection of hyperparameters for each model, the Optuna framework was employed to perform an optimization search, identifying the variable combinations that optimized each embedding model individually.

Each embedding set was then used by the baseline model to compute the recommendations. In this model, the average of the embeddings from each session was calculated, resulting in a single representative vector. The single vector was used to compute the cosine similarity to all known item embeddings, and the 10 most similar songs were recommended in descending order of similarity. The performance of this model was evaluated by using the metrics described in Section III-D.

Since the Optuna framework requires a performance metric to guide the hyperparameter search, the Hit Rate obtained from the baseline model was used as the objective function. This allowed the optimization process to identify the best hyperparameter combinations for each embedding configuration.

The best configuration found in each case is summarized in Table I. These settings are later used to train the final embeddings used by the recommendation models.

In Table I, we can see that some variants achieved the best performance with larger embedding dimensions. Although some models performed better with a vector dimensionality of 1024, these configurations were not used to train the neural network recommenders. Due to hardware limitations, particularly GPU memory constraints, embedding vectors of such high dimensionality exceeded the available capacity. Therefore, in each case where the optimal configuration included a 1024-dimensional embedding, the best-performing alternative with a lower dimensionality was chosen. For the Music4All dataset, the second-best embedding dimensionality was 256, while for the Xiami dataset, it was 512. This trade-off ensured model compatibility with the available resources while preserving the quality of the learned representations as much as possible.

The optimal context window size also varied between the datasets. For Xiami, larger window sizes yielded better results,

TABLE I
BEST HYPERPARAMETERS FOUND FOR EACH DATASET AND WORD2VEC VARIANT.

Dataset	Model	Vector Size	Window	Alpha	Min Alpha	Negative	Sample	HS
Music4All	Skip-Gram	512	7	0.0636	0.0007	13	0.0005	1
Music4All	CBOW	1024	9	0.0077	0.0028	19	0.0002	1
Xiami	Skip-Gram	64	15	0.0380	0.0013	5	0.0010	1
Xiami	CBOW	1024	8	0.0130	0.0027	18	0.0004	1

especially for Skip-Gram. This suggests that broader context information is beneficial in this dataset. In contrast, Music4All favored small to medium windows, suggesting that shorter context sequences may better reflect user preferences in this domain.

With the embeddings created and the baseline calculated, we then trained the LSTM and Transformer-based models using the same pre-trained embeddings. The architecture of these networks was designed so that the primary constraint on model complexity was the available GPU memory (VRAM). During experimentation, it was observed that increasingly complex models tended to perform better; however, at a certain point, further architectural elaboration became infeasible due to hardware limitations. In addition, embeddings with 1024 dimensions consumed a significant amount of memory, which prevented the use of deeper Transformer models and prevented effective training in these cases.

The LSTM and Transformer models are configured as shown in Tables II and III, respectively, with the architecture optimized to achieve the best possible performance for each combination of dataset and Word2Vec variant. Hyperparameters were tuned to ensure a fair comparison between the LSTM and Transformer models. As a result, both architectures were configured to have approximately the same model size for each combination of dataset and embedding dimension. Additionally, larger embedding sizes consistently required a greater number of neurons to train the model effectively, which is clearly reflected in Table II. A similar pattern was observed in the number of Transformer blocks: for models using 512-dimensional embeddings, four blocks were necessary to adequately capture the underlying information in the data.

Finally, we were able to compare the results of each model against the baseline. Table IV summarizes the evaluation metrics for the baseline, LSTM, and Transformer models across all datasets and embedding configurations.

With these results, we observe a consistent and significant improvement in all Transformer models compared to the baseline, and a modest yet noticeable improvement over the LSTM models. The Transformer achieved gains of up to 32.4% over the baseline and up to 6.2% over the LSTM model, highlighting its superior ability to combine session embedding vectors and generate more accurate predictions. A visual comparison is presented in Figure 4, reinforcing the Transformer's overall superiority while also illustrating its close performance to the LSTM variant. However, Transformer models tend to show a reduction in coverage, indicating a narrower diversity in the recommended items. Despite this,

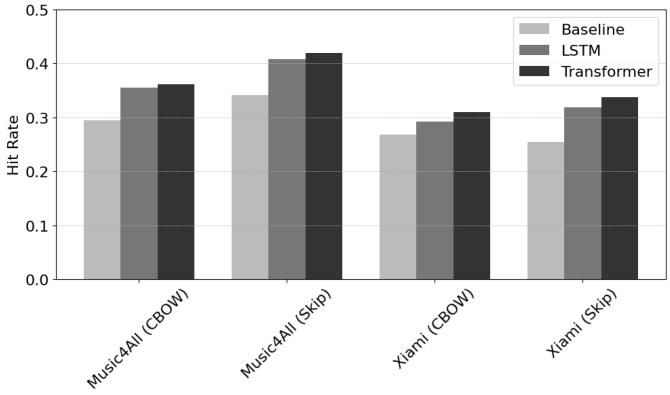

Fig. 4. Hit Rate by recommendation model, and combination of dataset and embedding model

Popularity Bias does not exhibit a consistent pattern across the evaluated models. It is worth noting that the Transformer model required a longer training time due to its increased complexity and a greater number of hyperparameters, making the overall tuning process more time-consuming.

In addition, the size of the Transformer models was limited by the computational resources available. There are indications that these results could be further improved with more refined and robust models, provided sufficient resources are available for training and experimentation. In addition, the Transformer and the LSTM models were roughly the same size for each database and embedding, allowing a fair comparison between them.

V. CONCLUSION AND FUTURE WORK

The goal of this work was to investigate whether Transformer-based architectures are capable of generating effective recommendations when operating on pre-trained embeddings and relying on cosine distance for item selection. The approach focused on encoding user sessions as sequences of vectors and predicting a representation that approximates the next item in the sequence. The results showed that Transformer models outperform the LSTM and baseline models in terms of Hit Rate, achieving higher accuracy in predicting the next song in user sessions. However, the opposite effect was observed in terms of Coverage: in three out of four cases, the Transformer models resulted in lower item coverage compared to the alternatives. Additionally, performance was ultimately constrained by the available computational resources, partic-

TABLE II
LSTM CONFIGURATION USED FOR ALL DATASETS AND WORD2VEC VARIANTS.

Dataset	Word2Vec Model	Embedding Dim.	Neurons	Dropout Rate
Music4All	Skip-Gram	512	1536	0.1
Music4All	CBOW	256	512	0.1
Xiami	Skip-Gram	64	256	0.1
Xiami	CBOW	512	2048	0.1

TABLE III
TRANSFORMER CONFIGURATION USED FOR ALL DATASETS AND WORD2VEC VARIANTS.

Dataset	Word2Vec Model	Embedding Dim.	Transformer Blocks	Attention Heads	FF Dim.	Dropout Rate
Music4All	Skip-Gram	512	4	2	1024	0.1
Music4All	CBOW	256	2	2	128	0.1
Xiami	Skip-Gram	64	2	4	512	0.1
Xiami	CBOW	512	4	2	4096	0.1

TABLE IV
COMPARING THE TRANSFORMER MODELS AGAINST TO THE LSTM MODEL AND THE BASELINE.

Dataset	Model	Hit Rate	Precision	Recall	F1 Score	MRR	NDCG	MAP	Coverage	Pop Bias
Music4All (CBOW)	Baseline	0.2949	0.0295	0.2949	0.0536	0.1148	0.1565	0.1148	0.8172	0.0339
Music4All (CBOW)	LSTM	0.3549	0.0355	0.3549	0.0645	**0.2219**	**0.2534**	**0.2219**	**0.8279**	0.0262
Music4All (CBOW)	Transformer	**0.3619**	**0.0362**	**0.3619**	**0.0658**	0.2048	0.2420	0.2048	0.7858	**0.0260**
Music4All (Skip)	Baseline	0.3412	0.0341	0.3412	0.0620	0.1210	0.1716	0.1210	**0.9981**	0.0280
Music4All (Skip)	LSTM	0.4081	0.0408	0.4081	0.0742	**0.2300**	**0.2723**	**0.2300**	0.9331	**0.0279**
Music4All (Skip)	Transformer	**0.4195**	**0.0420**	**0.4195**	**0.0763**	0.1983	0.2504	0.1983	0.9330	0.0304
Xiami (CBOW)	Baseline	0.2688	0.0269	0.2688	0.0489	0.1628	0.1876	0.1628	0.6131	0.0058
Xiami (CBOW)	LSTM	0.2924	0.0292	0.2924	0.0532	**0.2022**	**0.2236**	**0.2022**	0.6070	**0.0032**
Xiami (CBOW)	Transformer	**0.3104**	**0.0310**	**0.3104**	**0.0564**	0.1954	0.2225	0.1954	**0.6397**	0.0034
Xiami (Skip)	Baseline	0.2547	0.0255	0.2547	0.0463	0.1572	0.1801	0.1572	**0.7830**	**0.0028**
Xiami (Skip)	LSTM	0.3182	0.0318	0.3182	0.0579	0.2181	0.2418	0.2181	0.7458	0.0032
Xiami (Skip)	Transformer	**0.3373**	**0.0337**	**0.3373**	**0.0613**	**0.2330**	**0.2577**	**0.2330**	0.6577	**0.0028**

ularly the limitations of GPU memory, which restricted the complexity and dimensionality of the models.

For future work, we intend to conduct a direct comparison between the cosine-based method presented here and alternative architectures that predict items directly, thereby bypassing the embedding similarity step. In addition, we plan to explore the application of this methodology in other recommendation domains beyond music, such as e-commerce, news, or video streaming, providing further insight into its generalizability and effectiveness.

REFERENCES

[1] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, 2013.

[2] D. Wang, S. Deng, and G. Xu, "Sequence-based context-aware music recommendation," *Information Retrieval Journal*, vol. 21, no. 2–3, pp. 230–252, Oct. 2017, doi: https://doi.org/10.1007/s10791-017-9317-7.

[3] D. Jannach, M. Quadrana, and P. Cremonesi, "Session-based recommender systems," in *Recommender Systems Handbook*, 3rd ed., F. Ricci et al., Eds. Springer, 2022, pp. 301–335.

[4] S. Latifi, N. Mauro, and D. Jannach, "Session-aware recommendation: A surprising quest for the state-of-the-art," *Information Sciences*, vol. 573, pp. 291–315, Sep. 2021, doi: https://doi.org/10.1016/j.ins.2021.05.048.

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

[6] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," *arXiv preprint* arXiv:1606.08117, 2016. Available: https://arxiv.org/abs/1606.08117.

[7] A. Vaswani et al., "Attention is all you need," *arXiv preprint* arXiv:1706.03762, Jun. 2017. Available: https://arxiv.org/abs/1706.03762.

[8] L. Anido-Rifón et al., "Recommender systems," in *Re-engineering the Uptake of ICT in Schools*, Springer, 2015, pp. 91–114, doi: https://doi.org/10.1007/978-3-319-19366-3$_6$.

[9] D. Jannach and M. Zanker, "Value and impact of recommender systems," in *Recommender Systems Handbook*, Springer, 2012, pp. 519–546, doi: https://doi.org/10.1007/978-1-0716-2197-4$_1$4.

[10] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, Aug. 2019, doi: https://doi.org/10.3389/fams.2019.00044.

[11] S. Wang et al., "A survey on session-based recommender systems," *arXiv preprint* arXiv:1902.04864, Dec. 2020. Available: https://arxiv.org/abs/1902.04864.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint* arXiv:1301.3781, Sep. 2013. Available: https://arxiv.org/abs/1301.3781.

[13] O. Barkan and N. Koenigstein, "Item2Vec: Neural item embedding for collaborative filtering," *arXiv preprint* arXiv:1603.04259, 2016. Available: https://arxiv.org/abs/1603.04259.

[14] F. Sun et al., "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," *arXiv preprint* arXiv:1904.06690, Apr. 2019, doi: https://doi.org/10.48550/arXiv.1904.06690.

[15] I. A. Pegoraro Santana et al., "Music4All: A new music database and its applications," in *IEEE Xplore*, Jul. 2020. Available: https://ieeexplore.ieee.org/document/9145170.

[16] T. Akiba et al., "Optuna: A next-generation hyperparameter optimization

framework," *arXiv preprint* arXiv:1907.10902, Jul. 2019. Available: https://arxiv.org/abs/1907.10902.

[17] M. Douze et al., "The Faiss library," *arXiv preprint* arXiv:2401.08281, Jan. 2024. Available: https://arxiv.org/abs/2401.08281.

[18] J. L. Herlocker et al., "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, Jan. 2004, doi: https://doi.org/10.1145/963770.963772.

[19] Y. Chung et al., "Improved neighborhood search for collaborative filtering," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 1, pp. 29–40, Mar. 2018, doi: https://doi.org/10.5391/ijfis.2018.18.1.29.

[20] E. M. Voorhees and D. M. Tice, "The TREC-8 question answering track evaluation," in *Proceedings of the 8th Text Retrieval Conference (TREC-8)*, Gaithersburg, MD, Nov. 1999, pp. 77–82.

[21] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, Oct. 2002, doi: https://doi.org/10.1145/582415.582418.

[22] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: Evaluating recommender systems by coverage and serendipity," in *Proceedings of the ACM Conference on Recommender Systems*, Sep. 2010, doi: https://doi.org/10.1145/1864708.1864761.

[23] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the 11th ACM Conference on Recommender Systems*, Aug. 2017, doi: https://doi.org/10.1145/3109859.3109912.