MUSE: Music Recommender System with Shuffle Play Recommendation Enhancement

Yunhak Oh*
KAIST
Daejeon, Republic of Korea
yunhak.oh@kaist.ac.kr

Sukwon Yun*
KAIST
Daejeon, Republic of Korea
swyun@kaist.ac.kr

Dongmin Hyun
POSTECH
Pohang, Republic of Korea
dm.hyun@postech.ac.kr

Sein Kim
KAIST
Daejeon, Republic of Korea
rlatpdlsgns@kaist.ac.kr

Chanyoung Park†
KAIST
Daejeon, Republic of Korea
cy.park@kaist.ac.kr

ABSTRACT

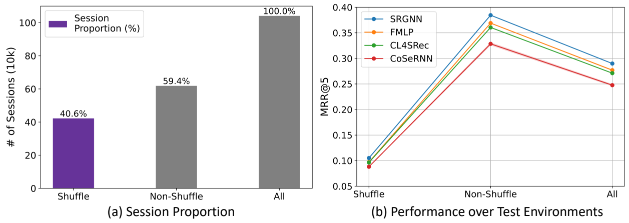
Recommender systems have become indispensable in music streaming services, enhancing user experiences by personalizing playlists and facilitating the serendipitous discovery of new music. However, the existing recommender systems overlook the unique challenges inherent in the music domain, specifically shuffle play, which provides subsequent tracks in a random sequence. Based on our observation that the shuffle play sessions hinder the overall training process of music recommender systems mainly due to the high unique transition rates of shuffle play sessions, we propose a **Mu**sic Recommender System with **S**huffle Play Recommendation **E**nhancement (MUSE). MUSE employs the self-supervised learning framework that maximizes the agreement between the original session and the augmented session, which is augmented by our novel session augmentation method, called transition-based augmentation. To further facilitate the alignment of the representations between the two views, we devise two fine-grained matching strategies, i.e., item- and similarity-based matching strategies. Through rigorous experiments conducted across diverse environments, we demonstrate MUSE's efficacy over 12 baseline models on a large-scale Music Streaming Sessions Dataset (MSSD) from Spotify. The source code of MUSE is available at https://github.com/yunhak0/MUSE.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

session-based recommendation, music recommendation, self-supervised learning



(a) Session Proportion

(b) Performance over Test Environments

Figure 1: Recommendation performance (MRR@5) of SBR models and music recommender model on MSSD-5d dataset.

ACM Reference Format:
Yunhak Oh, Sukwon Yun, Dongmin Hyun, Sein Kim, and Chanyoung Park. 2023. MUSE: Music Recommender System with Shuffle Play Recommendation Enhancement. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3614976

1 INTRODUCTION

Recommender systems [23, 25, 30, 32, 42, 44, 46] play a crucial role in providing an immersive user experience that navigates users to access various online content. Specifically, the recent prominence of Session-based Recommendation (SBR) lies in its ability to leverage implicit feedback gathered during a user's session, i.e., activities within a specified period. Due to their ability to adeptly handle session information, these applications have permeated our daily lives, with examples found across a range of domains, from books [3, 10], apparel [24, 38], and movies [2, 14, 28].

In contrast to such widely researched domains (e.g., books, fashion, or movies), building a successful music recommender system is especially challenging due to the inherent characteristics of the music domain, such as dependency on contextual factors, e.g., time or device user interacted, and rapid dynamics of user's interest. A few recent studies have aimed to alleviate such difficulties in the music domain. Hansen et al. [20] reflected past consumption and contextual factors (e.g., the time of the day, the device used to access the service, and stream sources). Moreover, Fazelnia et al. [16] recently proposed utilizing user representations that consider long-term, stable interests and rapidly shifting current preferences. Despite their progress in providing more personalized experiences, they overlook the prominent and essential characteristic that uniquely appears in the music domain: the *shuffle play* environments, where a set of tracks within a session are randomly provided.

*Both authors contributed equally to this research.
†Corresponding author.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom.

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614976

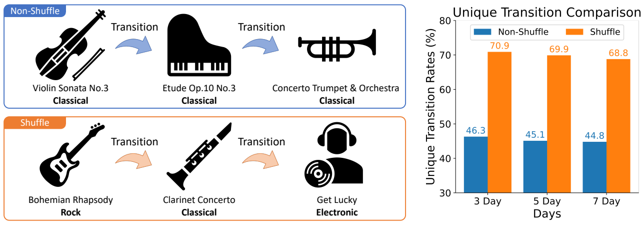


Figure 2: Comparison of unique transition rates between non-shuffle and shuffle play sessions in MSSD dataset. Unique transition rates = $\frac{\text{\# of Unique Transitions}}{\text{\# of Total Transitions}}$ (%)

However, shuffle play sessions should not be overlooked as they take a substantial proportion (i.e., 40.2%) of the total sessions (See Figure 1 (a)), implying that the shuffle play service is highly preferred by a large number of users and is frequently utilized in real-world scenarios. Moreover, an appropriate recommendation of a new track in shuffle play sessions would mitigate listening monotony and present serendipity in the user's auditory journey [29]. This perspective is corroborated by Spotify's "Smart Shuffle[1]," which is a recently launched service that aims to provide personalized shuffled sessions.

This work provides accurate recommendations for shuffle play sessions in the music domain. To begin with, we validate the effectiveness of existing state-of-the-art SBR models in each test environment. More precisely, we train SBR models (i.e., SRGNN [44], FMLP [51], and CL4SRec [46]) and a recent music recommender system (i.e., CoSeRNN [20]) on Music Streaming Sessions Dataset (MSSD) [9] provided by Spotify. We then evaluate their performance in detail in each test environment (as depicted in Figure 1 (b)). Notably, although the recommender models encountered numerous shuffle play sessions during training, they all performed poorly in predicting the next track in such sessions. Providing a satisfying recommendation in shuffle play sessions is extremely challenging compared to non-shuffle play sessions.

Then, an important question arises: Why do shuffle play sessions act as a bottleneck in building effective music recommender systems? The main clue lies in the difference in the music transition patterns between non-shuffle and shuffle play sessions. More precisely, as non-shuffle play sessions are rooted in a user's sequential history that reflects the user's taste, transitions within these sessions are unlikely to undergo dramatic shifts. For instance, if a user prefers classical music, the transitions within the session would be around similar classical music tracks. As a result, the number of *unique transitions*[2] would be small. However, in shuffle play sessions, where the next music is randomly provided to users, the number of unique transitions would be large compared to those of the non-shuffle case, as illustrated in Figure 2. Specifically, the unique transition rate in shuffle play sessions is considerably higher—about 1.5 times—than that of non-shuffle play sessions. Hence, we argue that such a unique transition poses a significant challenge for training SBR models, as the models need to accommodate rare and previously unseen transition patterns during training.

To this end, we propose a novel framework for training SBR, named **Mu**sic Recommender System with **S**huffle Play Recommendation **E**nhancement (MUSE), specifically designed to tackle the inherent challenges posed by shuffle play sessions in the music recommendation. MUSE captures the potential sequential information from shuffle play sessions using a novel session augmentation method, called transition-based augmentation. The main idea is to insert more frequently appearing transitions to reduce a considerable proportion of unique transitions, which results in more effective use of shuffle play sessions. Moreover, to obtain a robust unified encoder that works within diverse environments, we employ another augmentation method called reorder-based augmentation for non-shuffle play sessions, whose main idea is to mimic the shuffle-play environment.

After applying augmentations on shuffle and non-shuffle play sessions, we employ a self-supervised learning framework to maximize the agreement between the original and the augmented sessions. To further facilitate the alignment of representations between the two views, we introduce two fine-grained matching strategies, i.e., the item-based matching strategy that allows the identical items between the two views to be close in the embedding space, and the similarity-based matching strategy that supplements the alignment of similar embeddings between the views based on the nearest neighbors of each track.

Through extensive experiments, we demonstrate that MUSE outperforms recent SBR models and existing music recommender systems in predicting the next track, evaluated under various settings. To the best of our knowledge, this is the first work that attempts to enhance prevailing shuffle-play environments in the music domain in terms of training and inference.

In summary, our contributions are three-fold:

- We study the characteristic of the shuffle play sessions in the music recommender system and find that a large portion of unique transitions within shuffle play sessions poses a significant challenge for training existing SBR models. To this end, we propose a novel session augmentation method, called transition-based augmentation, that reduces the proportion of unique transitions of the shuffle play sessions.
- Our proposed method, MUSE, employs self-supervised learning to maximize the agreement between the original and augmented sessions. To further facilitate the alignment of the representations between the two views, we devise two fine-grained matching strategies, i.e., item- and similarity-based matching strategies.
- Through extensive experiments, we demonstrate the superiority of MUSE over recent session-based recommender models and a music recommender model in the next track prediction task in a real-world music streaming dataset, MSSD.

2 RELATED WORK

Session-based Recommendation (SBR). Recurrent Neural Networks (RNNs) have found utility in session-based recommendation (SBR), leveraging their capability to model sequential data. For instance, Hidasi et al. [22, 23] adapted the Gated Recurrent Unit (GRU) with the ranking loss function, aiming to predict the subsequent

[1] https://support.spotify.com/us/article/shuffle-play/

[2] A unique transition indicates a transition between tracks that appears only once.

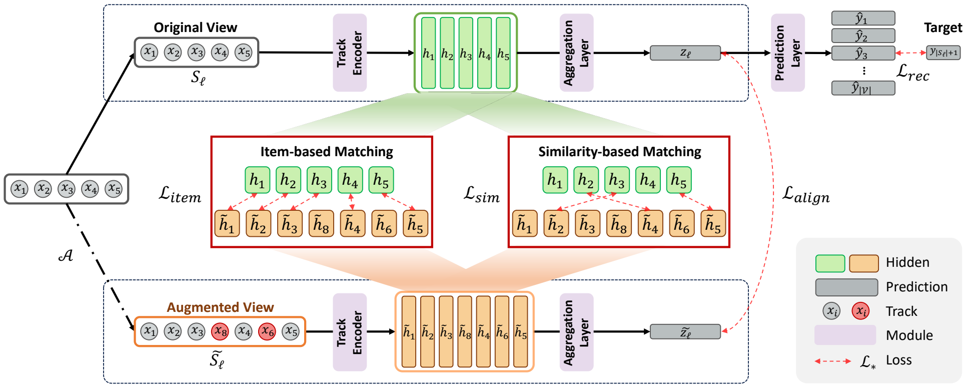


Figure 3: Overall architecture of MUSE. Given a session, we first generate an augmented view via transition-based augmentation in order to alleviate the unique transition problem. Subsequently, we employ item- and similarity-based matching to obtain a robust and unified encoder that can handle both shuffle and non-shuffle play sessions.

item of a session in SBR. NARM [30] extended GRU4Rec by incorporating an attention mechanism to capture the user's main purpose in the current session. Inspired by promising results in the Natural Language Processing (NLP) domain [41], some SBR models have embraced the self-attention mechanism. For example, SASRec [25] adopted the self-attention mechanism to capture both local and global interests. Recently, Graph Neural Networks (GNNs) have been proposed to derive item embedding for SBR using their ability to encode the relation between nodes. In particular, SR-GNN [44] leveraged gated GNN [31] to update item embeddings, considering complex item transitions, and it employed the attention mechanism akin to NARM's. Meanwhile, a few SBR models have focused on specific problems of implicit feedback, such as the noise of the sequential data [51]. FMLP [51] incorporated filtering algorithms from signal processing to minimize the noise in a session. Despite these advancements, none of these approaches adequately address the specific challenges associated with the music domain, such as the existence of shuffle play sessions. In contrast, our proposed model is designed to handle shuffle play sessions using item-matching and similarity-matching modules with a self-supervised learning framework. To our knowledge, this work is the first to explicitly address the challenges associated with shuffle play sessions.

Music Recommendation. In the music domain, the primary objective of the recommender system is to enrich the user experience by suggesting relevant tracks or artists that align with users' preferences. While achieving this goal, the discrepancy between industrial applications and academic research has been magnified due to the industry's exclusive access to online streaming data via their platforms. To close the gap, Spotify has partially released the Music Streaming Sessions Dataset (MSSD) [9] and even hosted a sequential skip prediction challenge[3]. It has led to numerous studies [1, 8, 11, 19, 34, 52] aiming to make better use of implicit user feedback, i.e., skips, to enhance user experience. However, these studies primarily concentrate on the skip prediction task. This task is a binary classification that operates under the assumption of having a set of items users are certain to consume in the near future, rendering it a relatively simple task. Although a few research [16, 20] has attempted to precisely predict the subsequent item a user will interact with, its broad impact is rather limited due to restricted access to the comprehensive dataset that includes user demographics or exact timestamps. Moreover, they overlook the shuffle play environments, which frequently co-occur with non-shuffle plays but negatively impact the overall training of the recommender system due to the inherent randomness involved. In this regard, we propose a novel framework for training SBR for the music domain that predicts the next track while considering shuffle play environments, arguably a complex and challenging task.

Self-supervised Learning (SSL). SSL has recently shown remarkable performance across various domains, including Computer Vision (CV) [5, 6, 12, 13, 18, 21, 49], Natural Language Processing (NLP) [15, 17, 45], and recommender systems [37, 46, 50]. SSL is a representation learning method that leverages supervision signals intrinsically generated from the data, eliminating the dependency on human-provided labels. Specifically, CL4SRec [46] proposed three data-level augmentations for the item sequence data and applied contrastive learning to enhance the user representation. Furthermore, DuoRec [37] suggested a model-level augmentation based on dropout [40] and applied supervised contrastive loss [26] as a regularizer to alleviate the representation degeneration problem. Unlike these approaches, mainly aiming at music recommender systems, our work proposes a novel data-level augmentation method that reflects the nature of the music domain.

3 PROPOSED FRAMEWORK: MUSE

In this section, we first formulate the problem of session-based recommendation (SBR) and self-supervised learning framework in Section 3.1. Then, we describe the architecture of MUSE. Specifically, in Section 3.2, we propose a novel session augmentation method for enhancing the robustness of the session encoder to shuffle-play sessions. In Section 3.3, we introduce fine-grained matching strategies between the original and augmented sessions, followed by the description of the aggregation and prediction layer in Section 3.4. Finally, we summarize the overall training process in Section 3.5.

3.1 Preliminaries

3.1.1 ***Problem Statement***. The objective of SBR is to predict a user's future interactions, specifically the subsequent track (i.e.,

[3] https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge

item). Given a session index ℓ with N sessions in total, a session $S_\ell = [x_1, x_2, ..., x_{|S_\ell|}]$ is composed of a sequence of tracks, where $x_t \in \mathcal{V}$ is the t-th track in the session and $\mathcal{V}$ is the set of all tracks in the data. The goal of is to predict the next track (i.e., $x_{|S_\ell|+1}$) given the past interactions $[x_1, \dots, x_{|S_\ell|}]$ in a given session S_ℓ. We aim to recommend top-K tracks for each session, given that user identity information is inaccessible due to the inherent nature of anonymous sessions.

3.1.2 ***Session-based Recommendation.*** Given an input session $S_\ell = [x_1, x_2, ..., x_{|S_\ell|}]$, recommender systems generally embed the tracks into embedding vectors, $\mathbf{E}_\ell = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{|S_\ell|}]$, where $\mathbf{e}_t \in \mathbb{R}^d$ is the d-dimensional embedding of the t-th track. Then, a track encoder f produces the representation of each track, $\mathbf{H}_\ell = f(\mathbf{E}_\ell) = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{|S_\ell|}]$, where $\mathbf{h}_t \in \mathbb{R}^d$, by modeling the interaction among tracks. Then, an aggregation layer g aggregates the track representations into a session representation $\mathbf{z}_\ell = g(\mathbf{H}_\ell)$ where $\mathbf{z}_\ell \in \mathbb{R}^d$. Given the session representation $\mathbf{z}$, a prediction layer with softmax operation produces the prediction probability for all tracks, $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_{|\mathcal{V}|}\}$. The training loss $\mathcal{L}_{rec}$ can be a classification loss such as cross-entropy loss. Lastly, the model recommends top-K tracks based on the prediction probability $\hat{\mathbf{y}}$.

3.1.3 ***Self-Supervised Learning (SSL) Framework.*** To leverage shuffle-play sessions during training, we propose a SSL framework, as shown in Figure 3. The framework takes a session S_ℓ as input, which can be either a shuffle or non-shuffle play session. Given the input session S_ℓ, an augmentation operation $\mathcal{A}$ augments the input session based on the transition frequency. As a result, the recommender system better captures users' preferences from the shuffle play sessions to provide more accurate recommendations. More formally, we embed the tracks into embedding vectors, $\mathbf{E}_\ell$ and $\tilde{\mathbf{E}}_\ell$, from the original and augmented sessions (i.e., S_ℓ and $\tilde{S}_\ell$), respectively. Then, a track encoder f produces track representations by modeling the interaction among the tracks in each session, i.e., $\mathbf{H}_\ell = f(\mathbf{E}_\ell)$ and $\tilde{\mathbf{H}}_\ell = f(\tilde{\mathbf{E}}_\ell)$. The aggregation layer g aggregates the track representations into session representations, i.e., $\mathbf{z}_\ell = g(\mathbf{H}_\ell)$ and $\tilde{\mathbf{z}}_\ell = g(\tilde{\mathbf{H}}_\ell)$. A basic SSL approach aligns the final representations (i.e.. $\mathbf{z}_\ell$ and $\tilde{\mathbf{z}}_\ell$) by increasing their similarity, resulting in the alignment loss (i.e., $\mathcal{L}_{align}$). We note that we employ a shared track encoder f and a shared aggregation layer g in both branches.

3.2 Transition-based Augmentation

Transition-based augmentation aims to enrich the sequential information in a given shuffle play session. To this end, we consider the transition frequency between items from all the sessions as an essential criterion for distinguishing shuffle and non-shuffle play sessions, as shown in Figure 4. We first demonstrate how we obtain a transition matrix and propose a novel session augmentation method conducted based on the transition matrix.

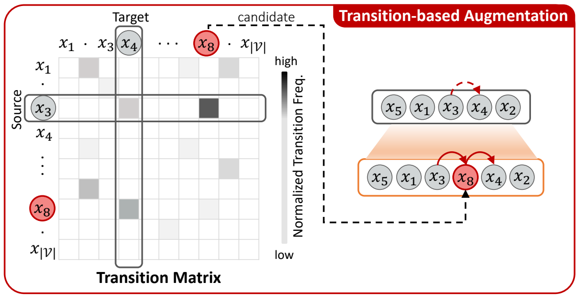

Transition-based Augmentation
Target
candidate
x_1 · x_3 x_4 · · · x_8 · $x_{|\mathcal{V}|}$
Source
x_1
x_3
x_4
x_8
$x_{|\mathcal{V}|}$
high
Normalized Transition Freq.
low
x_5 x_1 x_3 x_4 x_2
x_5 x_1 x_3 x_8 x_4 x_2
Transition Matrix

Figure 4: Our proposed transition-based augmentation showing an example of inserting a track x_8 between x_3 and x_4.

Transition Matrix. As shown in Figure 2, the main challenge inherent in the shuffle play sessions is their excessive amount of unique transitions within a session. To address the problem of excessive unique transitions, we introduce non-unique transition patterns observed across all sessions to shuffle play sessions. By doing so, we effectively mitigate the unique transition patterns inherent in shuffle play sessions, thereby unlocking the potential for leveraging these sessions during the training process. More formally, we first generate a transition frequency matrix $\mathbf{T} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, by collecting all transitions observed in the entire sessions as follows:

$$\mathbf{T}_{i,j} = \sum_{\ell=1}^{N} \sum_{t=1}^{|S_\ell|-1} \mathbb{1}([x_t, x_{t+1}] = [x_i, x_j]), \quad \forall i, j \leq |\mathcal{V}| \tag{1}$$

where $\mathbf{T}_{i,j}$ denotes the frequency of transition from source track x_i to target track x_j, $\mathbb{1}(a = b)$ denotes indicator function which outputs 1 if $a = b$ else 0, and N is the total number of sessions. We also take the logarithm to each value in $\mathbf{T}$ as the transition frequency of certain pairs, e.g., the transition between popular tracks, tends to be much higher than that of the remaining cases[4], which may incur the long-tail problem [27, 36, 39, 43, 48]. We then normalize the log-transformed matrix from the following two perspectives:

$$\bar{\mathbf{T}}_{i,\cdot} = \frac{\mathbf{T}_{i,\cdot}}{\sum_{j=1}^{|\mathcal{V}|} \mathbf{T}_{i,j}}, \quad \forall i \leq |\mathcal{V}|, \quad \bar{\mathbf{T}}_{\cdot,j} = \frac{\mathbf{T}_{\cdot,j}}{\sum_{i=1}^{|\mathcal{V}|} \mathbf{T}_{i,j}}, \quad \forall j \leq |\mathcal{V}| \tag{2}$$

where $\bar{\mathbf{T}}_{i,\cdot}$ and $\bar{\mathbf{T}}_{\cdot,j}$ denote the row-wise (i.e., source-wise) and column-wise (i.e., target-wise) normalized transition matrices, respectively. This results in the Markov Chain Transition Matrices [35], where the transition probability of each source and target node sums to one. This normalization enables us to interpret the transition matrix in terms of the probability distribution matrix and take a stochastic approach while augmenting a given session.

Transition-based Insertion. We now propose a novel session augmentation method to handle shuffle play sessions for music recommendation. The main idea is to insert frequently appearing transitions that could exist in a session. The primary goals of the augmentation are: (1) to reduce the excessive amount of unique transitions in shuffle play sessions and (2) to expose the session encoder to more diverse environments, thereby better accommodating shuffle play sessions. More precisely, our proposed augmentation method determines which items to be inserted at which locations in a given session. Here, the key idea lies in not inserting any random items but inserting relevant items that are likely to appear considering its back-and-forth context, i.e., source and target. For a clear and comprehensive understanding, the reader is encouraged to refer to Figure 4, which illustrates a toy example of inserting x_8 between x_3 and x_4. Specifically, as x_3 has a high transition probability to x_8, and x_8 has a high transition probability to x_4, we insert x_8 between x_3 and x_4.

[4] Here, we ensure log transformation is applied to non-zero frequency values.

For the efficiency of computation, we formally describe the insertion process of multiple tracks. Given an input session $S_\ell = [x_1, x_2, \dots, x_{|S_\ell|}]$, we have $|S_\ell| - 1$ candidate slots between the tracks for insertion. Thus, we set source tracks appearing before insertion (i.e., $S_\ell^{\mathrm{s}} = [x_1, x_2, \dots, x_{|S_\ell|-1}]$) and target tracks appearing after insertion (i.e., $S_\ell^{\mathrm{t}} = [x_2, x_3, \dots, x_{|S_\ell|}]$). Then, we obtain the transition matrices for source and target tracks such that:

$$\bar{\mathbf{T}}_{S_\ell^{\mathrm{s}},\cdot} \in \mathbb{R}^{(|S_\ell|-1)\times|\mathcal{V}|}, \quad \bar{\mathbf{T}}_{\cdot,S_\ell^{\mathrm{t}}} \in \mathbb{R}^{|\mathcal{V}|\times(|S_\ell|-1)}. \tag{3}$$

Figure 4 shows an example of insertion between x_3 and x_4, while Eq. 3 considers all cases of insertion. We then obtain the probability of candidate tracks for insertion between source and target tracks as follows:

$$\mathbf{P}_{S_\ell,\cdot} = \bar{\mathbf{T}}_{S_\ell^{\mathrm{s}},\cdot} \odot \bar{\mathbf{T}}^\top_{\cdot,S_\ell^{\mathrm{t}}} \tag{4}$$

where $\mathbf{P}_{S_\ell,\cdot} \in \mathbb{R}^{(|S_\ell|-1)\times|\mathcal{V}|}$ denotes a matrix of potential candidates that could be inserted between tracks in a given session, with values obtained via Hadamard product, $\odot$, of two subsets of transition probability matrices[5]. It is important to note that as both matrices consist of Markov Chain Transition probabilities, the potential candidates would contain a value that naturally considers its stochastic nature, conditioned on both the source track and the target track. We also apply row-wise softmax to ensure the probability distribution, i.e., $\bar{\mathbf{P}}_{S_\ell,\cdot} = \mathsf{softmax}(\bar{\mathbf{T}}_{S_\ell^{\mathrm{s}},\cdot} \odot \bar{\mathbf{T}}^\top_{\cdot,S_\ell^{\mathrm{t}}})$.

Based on the potential candidates obtained from transition matrices, we sample a candidate track to be inserted in each interval of the sequence as follows:

$$\mathbf{c}_i = \begin{cases} \mathsf{Multinomial}(\bar{\mathbf{P}}_{S_\ell}[i,:]), & \text{if } \mathsf{sum}(\bar{\mathbf{P}}_{S_\ell}[i,:]) > 0 \\ \varnothing, & \text{otherwise} \end{cases}, \forall i \le |S_\ell| - 1 \tag{5}$$

where $\mathbf{c}_i$ is the i-th element of $\mathbf{c} \in \mathbb{R}^{|S_\ell|-1}$, which is initialized as zeros then replaced with a sample obtained from Multinomial Distribution with event probabilities, $\bar{\mathbf{P}}_{S_\ell}[i,:]$, given at least one candidate, i.e., a potential track that is associated to both a transition from the source track and a transition to the target track exists ($\mathsf{sum}(\bar{\mathbf{P}}_{S_\ell}[i,:]) > 0$). An example of such a potential track is x_8 in Figure 4. Finally, we obtain the augmented session $\tilde{S}_\ell$ by inserting the sampled tracks $\mathbf{c}$ into the original session S_ℓ. Here, when new tracks are inserted between each track in the original session, we employ the augmented session unless its length surpasses the maximum session length. However, if the number of candidate tracks exceeds the available slots (i.e., $X - (|S_\ell| - 1)$, where X is the maximum session length), we randomly pick tracks from the candidates to ensure that each track has an equal opportunity for integration into the session. In summary, the augmented session alleviates the unique transition problem by inserting relevant transitions.

Discussions on non-shuffle play sessions. Heretofore, we mainly discussed augmenting shuffle play sessions. However, we can also benefit from applying augmentations to the non-shuffle play sessions, as shown by an existing work [46]. Here, we opted not to apply transition-based augmentation to non-shuffle play sessions, given that their transition patterns are not as unique as those of shuffle play sessions. Furthermore, we empirically observed that using transition-based augmentation for non-shuffle play sessions did not result in a performance gain. Instead, we apply reorder-based augmentation for non-shuffle play sessions, which randomly reorders the tracks within a session, thereby mimicking the shuffle play environment. By exposing the reordered non-shuffle play sessions to the session encoder, we obtain a robust and unified encoder invariant to the shuffles.

3.3 Item- and Similarity-based Matching

In this section, we propose fine-grained matching strategies, i.e., item- and similarity-based matching, to better align the original and augmented session. As illustrated in Figure 3, we obtain an augmented session $\tilde{S}_\ell$ of the input session S_ℓ through an augmentation operation $\mathcal{A}$. After looking up the embedding vectors $\mathbf{E}_\ell \in \mathbb{R}^{|S_\ell|\times d}$, and $\tilde{\mathbf{E}}_\ell \in \mathbb{R}^{|\tilde{S}_\ell|\times d}$ for each item within each view, the track encoder f generates track representations, i.e., $\mathbf{H}_\ell \in \mathbb{R}^{|S_\ell|\times d}$ and $\tilde{\mathbf{H}}_\ell \in \mathbb{R}^{|\tilde{S}_\ell|\times d}$, corresponding to each view. We now delineate the matching strategies.

3.3.1 *Item-based Matching.*

The augmentations make the encoder generate different hidden representations of the same items due to the differing adjacent items. Nonetheless, we aim to make the encoder to be invariant to such augmentations. The item-based matching ensures the alignment between the two views' hidden representations derived from the same items. Let the items from original session S_ℓ be $\mathbf{I}_\ell = \{x_i | x_i \in S_\ell\}$ and the items from augmented session $\tilde{S}_\ell$ be $\tilde{\mathbf{I}}_\ell = \{x_i | x_i \in \tilde{S}_\ell\}$. The item-based matching loss function, Mean Squared Error, is defined as follows:

$$\mathcal{L}_{item} = \frac{1}{|\mathbf{I}_\ell|} \sum_{x_t \in \mathbf{I}_\ell} \sum_{x_k \in \tilde{\mathbf{I}}_\ell} \mathbb{1}(x_t = x_k) \|\mathbf{h}_t - \tilde{\mathbf{h}}_k\|^2, \tag{6}$$

where $\mathbb{1}(a = b)$ is the indicator that produces 1 if $a = b$ and 0 otherwise, $\mathbf{h}_t, \tilde{\mathbf{h}}_k \in \mathbb{R}^d$ are representations of t-th track in the original session and the k-th track in the augmented session, respectively.

3.3.2 *Similarity-based Matching.*

In addition to the item-based matching strategy, inspired by the importance of neighborhood information in recommender systems [7], we employ similarity-based matching that aligns representations of similar items. Unlike item-based matching, which focuses on aligning the representations of the same item from the two views, similarity-based matching determines the nearest neighbor of each item in one view from the items in the other view. To accomplish this, we first calculate the Euclidean distance in the embedding space between all pairs of tracks in the original session S_ℓ and the augmented session $\tilde{S}_\ell$, then select the nearest neighbor (NN) track for each track representation as follows:

$$\mathcal{P}(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell) = \{\, (\mathbf{h}_i, \mathsf{NN}(\mathbf{h}_i, \tilde{\mathbf{H}}_\ell)) \,|\, \mathbf{h}_i \in \mathbf{H}_\ell \} \tag{7}$$

where $\mathcal{P}(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell)$ is the set of track pairs in which one is from the original session and the other is its nearest neighbor from the augmented session. $|\mathcal{P}(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell)| = |S_\ell|$, and $\mathsf{NN}(\mathbf{h}_i, \tilde{\mathbf{H}}_\ell)$ returns the representation of the nearest neighbor track of the i-th track of the original session (i.e., $\mathbf{h}_i$) among all tracks in the augmented session (i.e., $\tilde{\mathbf{H}}_\ell$). Then, we select the top-κ tracks with the lowest distance denoted as $\mathcal{P}^\kappa(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell)$, where $|\mathcal{P}^\kappa(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell)| = \kappa$ and $\mathcal{P}^\kappa(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell) \subset \mathcal{P}(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell)$. The purpose of introducing top-κ selection is to ensure that only similar pairs are considered so that

[5] For the implementation, transition matrix is stored as sparse tensors regarding its high sparsity, hence the memory cost is notably low.

this process complements the item-based matching. Likewise, we consider the top-κ nearest neighbors in the perspective of the augmented session, i.e., $\mathcal{P}^{\kappa}(\tilde{\mathbf{H}}_\ell, \mathbf{H}_\ell)$. The loss function for similarity-based matching is defined as follows:

$$\mathcal{L}_{sim} = \sum_{(\mathbf{h}_i, \mathsf{NN}(\mathbf{h}_i, \tilde{\mathbf{H}}_\ell)) \in \mathcal{P}^{\kappa}} \|\mathbf{h}_i - \mathsf{NN}(\mathbf{h}_i, \tilde{\mathbf{H}}_\ell)\|^2 + \sum_{(\tilde{\mathbf{h}}_i, \mathsf{NN}(\tilde{\mathbf{h}}_i, \mathbf{H}_\ell)) \in \tilde{\mathcal{P}}^{\kappa}} \|\tilde{\mathbf{h}}_i - \mathsf{NN}(\tilde{\mathbf{h}}_i, \mathbf{H}_\ell)\|^2 \quad (8)$$

where $\mathcal{P}^{\kappa} = \mathcal{P}^{\kappa}(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell)$ and $\tilde{\mathcal{P}}^{\kappa} = \tilde{\mathcal{P}}^{\kappa}(\tilde{\mathbf{H}}_\ell, \mathbf{H}_\ell)$ for simplicity. It is worth noting that incorporating similarity-based matching from the beginning may interfere with the training, as the representations are not yet established. Therefore, after some warm-up epochs with only the item-based matching, we start the similarity-based matching, aiming to obtain meaningful representations[6].

3.3.3 ***Regularization.*** To avoid the representation collapse problem prevalent in the self-supervised learning framework, we employ the regularization strategy introduced in VICReg [5]: $\mathcal{L}_{VICReg} = \lambda \cdot s(\mathbf{H}_\ell, \tilde{\mathbf{H}}_\ell) + \mu[v(\mathbf{H}_\ell) + v(\tilde{\mathbf{H}}_\ell)] + \nu[c(\mathbf{H}_\ell) + c(\tilde{\mathbf{H}}_\ell)]$, where s, v, and c are the invariance, variance, and covariance terms, respectively, and λ, μ, and ν are scalar coefficients terms. Therefore, the final loss function of these matching is defined as follows:

$$\mathcal{L}_{matching} = \mathcal{L}_{item} + \mathcal{L}_{sim} + \mathcal{L}_{VICReg}. \quad (9)$$

3.4 Aggregation and Prediction Layer

After the track encoder models all interactions among tracks in a session, we leverage the track representations $\mathbf{H}_\ell = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{|S_\ell|}]$ to aggregate both long-term preference and current interests of the session. In the aggregation layer g, We first consider the last track representation $\mathbf{h}_{|S_\ell|}$ as the local embedding $\mathbf{z}_\ell^{(\text{local})}$ of the session, i.e., $\mathbf{z}_\ell^{(\text{local})} = \mathbf{h}_{|S_\ell|}$. Then, we derive the global embedding $\mathbf{z}_\ell^{(\text{global})}$ from all track representations. Here we adopt Bahdanau attention [4] by following the previous works [30, 32, 44]:

$$\mathbf{z}_\ell^{(\text{global})} = \sum_{i}^{|S_\ell|} \beta_i \mathbf{h}_i, \;\; \beta_i = \mathbf{W}_1^T \sigma(\mathbf{W}_2 \mathbf{h}_i + \mathbf{W}_3 \mathbf{h}_{|S_\ell|} + \mathbf{b}) \quad (10)$$

where learnable parameters $\mathbf{W}_1 \in \mathbb{R}^d$, $\mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b}$ control the weight of track representations, and σ is an activation function. Finally, we concatenate and transform local $\mathbf{z}_\ell^{(\text{local})}$ and global $\mathbf{z}_\ell^{(\text{global})}$ embedding to a d-dimensional embedding: $\mathbf{z}_\ell = \mathbf{W}_4(\mathbf{z}_\ell^{(\text{local})} \oplus \mathbf{z}_\ell^{(\text{global})})$, where $\oplus$ is a concatenate operator and $\mathbf{W}_4 \in \mathbb{R}^{d \times 2d}$. After feeding the augmented track representation into this aggregation layer, we obtain the augmented session representation $\tilde{\mathbf{z}}_\ell = g(\tilde{\mathbf{H}}_\ell)$. To align the two session representations, we employ the self-supervised loss introduced in VICReg [5]:

$$\mathcal{L}_{align} = \lambda \cdot s(\mathbf{z}_\ell, \tilde{\mathbf{z}}_\ell) + \mu[v(\mathbf{z}_\ell) + v(\tilde{\mathbf{z}}_\ell)] + \nu[c(\mathbf{z}_\ell) + c(\tilde{\mathbf{z}}_\ell)]. \quad (11)$$

Given the session representation $\mathbf{z}$, a prediction layer computes the prediction probability $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{V}|}$ of the next track using softmax:

$$\hat{\mathbf{y}} = \mathsf{softmax}(\mathbf{z}_\ell^T \mathbf{e}_i),$$

where $\mathbf{e}_i \in \mathbf{E}_{\mathcal{V}}$ is a candidate item embedding vector. For each session, we minimize the cross entropy loss defined as follows:

$$\mathcal{L}_{rec} = -\sum_{i=1}^{|\mathcal{V}|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \quad (12)$$

where $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{V}|}$ is the one-hot vector of the target track.

Table 1: Statistics of datasets.

Statistics	MSSD-3d	MSSD-5d	MSSD-7d
# of plays	11,858,262	16,701,958	19,366,448
# of shuffle play sessions	301,814	422,221	501,875
# of non-shuffle play sessions	442,726	618,701	713,300
# of training sessions	613,308	909,818	1,061,274
# of test sessions	131,232	131,104	153,901
# of tracks	199,177	253,693	280,079
Average length	15.93	16.05	15.94

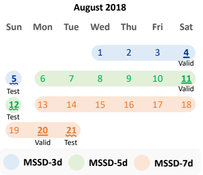


Figure 5: Days split.

3.5 Model Training

To sum up, the final loss of MUSE can be expressed as follows:

$$\mathcal{L}_{\text{final}} = \alpha \mathcal{L}_{matching} + (1 - \alpha)\mathcal{L}_{align} + \mathcal{L}_{rec}, \quad (13)$$

where α is a loss-controlling hyperparameter that balances between the matching loss and alignment loss, $\mathcal{L}_{matching}$ accounts for the item- and similarity-based matching loss with a regularization loss (Eq. 9), $\mathcal{L}_{align}$ aims for the alignment of embeddings obtained via the aggregation layer (Eq. 11), and $\mathcal{L}_{rec}$ is derived from the next track prediction task through the prediction layer (Eq. 12).

4 EXPERIMENTS

We first describe our experiment settings in Section 4.1 and summarize observations from the overall performance in Section 4.2. Additionally, we delineate the effectiveness of MUSE in non-shuffle and shuffle play environments in Section 4.3. In Section 4.4, we demonstrate the efficacy of our proposed module, e.g., transition-based augmentation and fine-grained matching strategies. Finally, we show the sensitivity of each hyperparameter in Section 4.5.

4.1 Experimental Settings

4.1.1 ***Dataset.*** We compare MUSE with baseline methods on a large-scale real-world dataset, Music Streaming Sessions Dataset (MSSD) [9], from Spotify[7]. It comprises 160 million listening sessions with 20 billion plays, accompanied by user actions. It consists of the historical logs for 66 days. Additionally, to ensure manageable computation time, we utilize about 50% of the entire dataset due to its extensive size[8]. We then constitute 3 chunks of the dataset by selecting data belonging to a few days as adopted in a conventional work [30] that used chunk data of original data due to its large size. As illustrated in Figure 5, here are the 3 chunks of the dataset:

- 3 days (MSSD-3d) - training data is from 1 August 2018 to 3 August 2018, validation data is from 4 August 2018, and test data is from 5 August 2018,
- 5 days (MSSD-5d) - training data is from 6 August 2018 to 10 August 2018, validation data is from 12 August 2018, and test data is from 13 August 2018
- 7 days (MSSD-7d) - training data is from 13 August 2018 to 20 August 2018, validation data is from 21 August 2018, and test data is from 22 August 2018.

[6] In experiments, we used the first epoch of the training as the warm-up epoch.

[7] https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge

[8] We used data files whose file names start with log_0, log_1, log_2, log_3, and log_4.

For the data preprocessing, we excluded items in the test data that do not appear in the training data, i.e., cold-start problem that is generally covered as a separate issue. We filtered out non-premium users because they are limited to using the streaming platform as done in [20]. Following the conventional works [23, 30], we also filtered out sessions containing only one track and tracks that appear less than 5 times in training data. In addition, for the session $S = [x_1, x_2, \ldots, x_{|S|}, x_{|S|+1}]$, we set up a series of sequences and corresponding labels $([x_1], x_2), ([x_1, x_2], x_3), \ldots, ([x_1, x_2, \ldots, x_{|S|}], x_{|S|+1})$, where $([*], \cdot)$ denotes a track sequence $[*]$ and next tracks $\cdot$. However, when generating a series of sequences and next tracks, we filtered out data instances that the user skipped the next tracks in order to recommend a track that a user will listen to. Specifically, for a shuffle play session, we excluded all skipped tracks, even in input. Because the shuffle play session inherits the randomness, we exclude them to construct more meaningful track sequences. For example, given that the session is $S_\ell^{(Shuffle)} = [x_1, x_2, x_3, x_4, x_5]$ and the user listening behavior is [*listen*, *skip*, *listen*, *skip*, *listen*], a series of sequences and corresponding labels are generated as follows: $([x_1], x_3), ([x_1, x_3], x_5)$. Lastly, if the shuffle play mode (e.g., shuffle play → non-shuffle play) is changed in the middle of a session, we treat it as a shuffle play session. The detailed statistics of the dataset after the preprocessing are in Table 1.

4.1.2 *Evaluation Protocol*.

Since music recommender systems typically present a limited number of tracks at a time, it is important that the actual track listened to by the user is included in the top-ranked tracks of the list. Therefore, we adopt the Recall (**Recall@K**) [23, 30], Mean Reciprocal Rank (**MRR@K**) [23, 30, 44], and Normalized Discounted Cumulative Gain (**NDCG@K**) [46, 51] that are frequently used when evaluating the ranking performance.

4.1.3 *Compared Methods*.

We evaluate our proposed method compared with the following baseline methods. **General Recommender System(RS)**: 1) SimpleX [33] is a collaborative filtering method with Cosine Contrastive Loss. We used average track embeddings as a user embedding of a session. **Classic SBR**: 2) GRU4Rec [23] is RNN-based SBR (i.e., GRU) with a ranking loss function to encode the sequential information. **Attention-based SBR**: 3) NARM [30] is an RNN-based model with an attention mechanism to aggregate long- and short-term interest. 4) STAMP [32] is an MLP-based model with a short-term attention priority module to detect shifts in user interest in a session. 5) CSRM [42] is an extended model of NARM with outer memory to utilize collaborative signals from neighbor sessions. 6) SASRec [25] utilized unidirectional transformer architecture. **Graph-based SBR**: 7) SR-GNN [44] utilized Gated GNN to encode the item embedding and aggregate them using an attention mechanism. 8) GC-SAN [47] is a method for the fusion of Gated GNN and self-attention mechanism. **Self-supervised Learning-based SBR**: 9) CL4SRec [46] adopted a contrastive learning framework with SASRec as a backbone. 10) DuoRec [37] utilized model-based augmentation with supervised contrastive loss. **Recent SBR**: FMLP [51] replaced the self-attention module with a simple MLP with a noise-filtering algorithm from signal processing. **Music RS**: CoSeRNN [20] is an RNN-based (i.e., LSTM) music recommender system that utilizes context information (e.g., day, stream source). We excluded some context variables that they used due to the data availability. We utilized context (e.g., charts, personalized playlists, user collection) and item embedding as learnable embedding; the other is used as one-hot embedding.

4.1.4 *Implementation Details*.

Hyperparameters Tuning. We tuned the hyperparameters of the methods, including MUSE based on MRR@5 on the validation dataset. We then evaluated the methods with the optimal hyperparameters on the test dataset when they produced the highest MRR@5 on the validation dataset.

Compared Methods. The maximum length of sessions is 20 tracks, which is given in MSSD. We add zero padding if a session contains fewer than 20 tracks. For fair comparisons, we set the hidden dimension d to 100 and batch size to 512. We also use a single layer for all graph neural networks and the transformer encoder layer. Additionally, we use a single head for the transformer encoder layer. The other model-specific hyperparameters were searched in the range reported by the authors.

Our Proposed Framework. For MUSE, we searched the reorder probability γ that controls the proportion of tracks in a given session in $\{0.3, 0.5, 0.7, 0.9\}$. For κ, which is responsible for selecting the top-κ nearest neighbor pairs in the similarity-based matching process, we fixed it as 5. We searched α in $\{0.2, 0.4, 0.6, 0.8\}$ (Eq. 13), which is the loss-controlling hyperparameter balancing the matching loss and alignment loss. For the coefficients used in VICReg regularization, we fixed λ, μ, and ν to 1, 1, and 10, respectively.

4.2 Overall Performance Comparison

To demonstrate the effectiveness of MUSE, we report the recommendation accuracy of MUSE and all the baselines in Table 2. MUSE aims to fully leverage the shuffle play sessions through transition-based augmentation and fine-grained matching strategies, i.e., item- and similarity-based matching. We summarize the following observations: 1) MUSE achieves state-of-the-art performance in the real-world, large-scale dataset (i.e., MSSD) over 12 baseline recommender systems, which demonstrates the superiority of MUSE. 2) CoSeRNN, a music recommender system, performs inferior to MUSE and other baseline models. We speculate that CoSeRNN is designed to depend heavily on user identity and contextual information (e.g., device type), while they are not provided in MSSD. In contrast, MUSE shows superior performance without the auxiliary information, which signifies the practicality of MUSE. 3) Graph-based methods, e.g., SRGNN and GCSAN, show the highest performance over the other baseline methods. The graph-based methods mainly utilize the transition between tracks in sessions by constructing graphs. Thus, it implies that transition information is important in music recommendation. MUSE also takes the ability of graph by taking SRGNN as the backbone. In addition to GNN, the transition-based augmentation further supplements the transition information into the shuffle-play sessions, which supports the superior performance of MUSE. 4) As self-supervised learning (SSL) approaches, e.g., CL4SRec and DuoRec, improve the performance of the backbone (i.e., SASRec), MUSE significantly outperforms SRGNN. It implies that the SSL framework fully utilizes the backbone's ability with the same number of parameters. 5)

Table 2: Overall performance comparison. Gen., Cls., and Rec. denote general, classic, and recent, respectively. Δ_b and Δ_s denote the relative improvement of MUSE over the backbone model, SRGNN, and state-of-the-art baseline, GCSAN, respectively. The performance is averaged across 5 log files in each chunk, and its standard deviation is shown in parentheses. Bold fonts indicate the top-ranking performance while underlining denotes the second-place performance. An asterisk (*) indicates the statistical significance of the improvement of our model over the top-performing baseline, as determined by a paired t-test with $p < 0.01$.

Setting		Gen. RS	Cls. SBR	Attention-based SBR				Graph-based SBR		SSL-based SBR		Rec. SBR	Music RS	**Ours**		
Dataset	Metric	SimpleX	GRU4Rec	NARM	STAMP	CSRM	SASRec	SRGNN	GCSAN	CL4SRec	DuoRec	FMLP	CoSeRNN	**MUSE**	Δ_b	Δ_s
MSSD 3d	R@5	0.1785 (0.0019)	0.2359 (0.0019)	0.3348 (0.0021)	0.3283 (0.0017)	0.3402 (0.0011)	0.3346 (0.0027)	0.3502 (0.0018)	0.3559 (0.0023)	0.3380 (0.0018)	0.3381 (0.0017)	0.3438 (0.0023)	0.3037 (0.0011)	**0.3628*** (0.0025)	3.60%	1.94%
	R@10	0.2982 (0.0014)	0.2753 (0.0019)	0.3882 (0.0026)	0.3795 (0.0019)	0.3944 (0.0029)	0.3897 (0.0037)	0.4013 (0.0022)	0.4058 (0.0030)	0.3939 (0.0024)	0.3931 (0.0021)	0.3986 (0.0023)	0.3648 (0.0013)	**0.4145*** (0.0029)	3.29%	2.14%
	M@5	0.0904 (0.0009)	0.1802 (0.0021)	0.2724 (0.0018)	0.2644 (0.0014)	0.2765 (0.0021)	0.2670 (0.0016)	0.2861 (0.0019)	0.2930 (0.0014)	0.2689 (0.0021)	0.2695 (0.0021)	0.2758 (0.0016)	0.2324 (0.0012)	**0.2974*** (0.0020)	3.95%	1.50%
	M@10	0.1061 (0.0009)	0.1854 (0.0020)	0.2795 (0.0018)	0.2712 (0.0014)	0.2837 (0.0019)	0.2743 (0.0016)	0.2929 (0.0019)	0.2996 (0.0015)	0.2763 (0.0020)	0.2768 (0.0020)	0.2831 (0.0016)	0.2404 (0.0012)	**0.3043*** (0.0020)	3.89%	1.57%
	N@5	0.1120 (0.0012)	0.1941 (0.0020)	0.2880 (0.0018)	0.2803 (0.0014)	0.2924 (0.0016)	0.2838 (0.0018)	0.3021 (0.0019)	0.3087 (0.0016)	0.2861 (0.0019)	0.2866 (0.0019)	0.2927 (0.0018)	0.2501 (0.0011)	**0.3137*** (0.0021)	3.84%	1.62%
	N@10	0.1505 (0.0010)	0.2068 (0.0020)	0.3052 (0.0018)	0.2968 (0.0015)	0.3099 (0.0012)	0.3016 (0.0019)	0.3186 (0.0018)	0.3248 (0.0019)	0.3041 (0.0019)	0.3043 (0.0017)	0.3104 (0.0018)	0.2698 (0.0011)	**0.3304*** (0.0022)	3.70%	1.72%
MSSD 5d	R@5	0.1712 (0.0009)	0.2329 (0.0010)	0.3394 (0.0016)	0.3316 (0.0014)	0.3440 (0.0013)	0.3350 (0.0017)	0.3529 (0.0010)	0.3562 (0.0012)	0.3352 (0.0016)	0.3378 (0.0020)	0.3438 (0.0012)	0.3159 (0.0020)	**0.3636*** (0.0005)	3.03%	2.08%
	R@10	0.2884 (0.0013)	0.2745 (0.0014)	0.3941 (0.0032)	0.3841 (0.0010)	0.3990 (0.0018)	0.3891 (0.0021)	0.4040 (0.0020)	0.4065 (0.0015)	0.3886 (0.0019)	0.3926 (0.0026)	0.3989 (0.0015)	0.3747 (0.0012)	**0.4153*** (0.0008)	2.80%	2.16%
	M@5	0.0872 (0.0006)	0.1745 (0.0004)	0.2764 (0.0005)	0.2671 (0.0014)	0.2804 (0.0013)	0.2701 (0.0014)	0.2899 (0.0007)	0.2939 (0.0011)	0.2711 (0.0010)	0.2717 (0.0015)	0.2769 (0.0013)	0.2476 (0.0023)	**0.2993*** (0.0006)	3.24%	1.84%
	M@10	0.1025 (0.0006)	0.1800 (0.0005)	0.2836 (0.0007)	0.2741 (0.0013)	0.2876 (0.0013)	0.2772 (0.0013)	0.2967 (0.0007)	0.3006 (0.0011)	0.2781 (0.0010)	0.2790 (0.0015)	0.2843 (0.0011)	0.2554 (0.0022)	**0.3062*** (0.0005)	3.20%	1.86%
	N@5	0.1078 (0.0006)	0.1890 (0.0005)	0.2920 (0.0008)	0.2832 (0.0013)	0.2962 (0.0012)	0.2863 (0.0014)	0.3056 (0.0006)	0.3094 (0.0011)	0.2870 (0.0011)	0.2882 (0.0016)	0.2936 (0.0011)	0.2646 (0.0022)	**0.3154*** (0.0005)	3.21%	1.94%
	N@10	0.1455 (0.0007)	0.2025 (0.0007)	0.3096 (0.0012)	0.3001 (0.0012)	0.3139 (0.0012)	0.3037 (0.0013)	0.3221 (0.0008)	0.3257 (0.0011)	0.3042 (0.0011)	0.3059 (0.0018)	0.3114 (0.0010)	0.2836 (0.0019)	**0.3320*** (0.0004)	3.07%	1.93%
MSSD 7d	R@5	0.1749 (0.0020)	0.2259 (0.0019)	0.3363 (0.0013)	0.3257 (0.0007)	0.3388 (0.0009)	0.3314 (0.0019)	0.3498 (0.0014)	0.3522 (0.0013)	0.3336 (0.0011)	0.3344 (0.0009)	0.3401 (0.0017)	0.3086 (0.0031)	**0.3607*** (0.0018)	3.12%	2.41%
	R@10	0.2903 (0.0018)	0.2699 (0.0020)	0.3943 (0.0022)	0.3803 (0.0006)	0.3960 (0.0012)	0.3887 (0.0029)	0.4038 (0.0016)	0.4054 (0.0016)	0.3906 (0.0012)	0.3917 (0.0009)	0.3979 (0.0020)	0.3695 (0.0028)	**0.4150*** (0.0024)	2.77%	2.37%
	M@5	0.0898 (0.0011)	0.1653 (0.0018)	0.2690 (0.0006)	0.2584 (0.0011)	0.2732 (0.0006)	0.2625 (0.0008)	0.2832 (0.0006)	0.2860 (0.0007)	0.2647 (0.0012)	0.2650 (0.0008)	0.2698 (0.0011)	0.2394 (0.0030)	**0.2929*** (0.0012)	3.43%	2.41%
	M@10	0.1049 (0.0011)	0.1711 (0.0018)	0.2766 (0.0006)	0.2656 (0.0010)	0.2807 (0.0006)	0.2701 (0.0008)	0.2904 (0.0007)	0.2931 (0.0006)	0.2722 (0.0012)	0.2725 (0.0008)	0.2775 (0.0011)	0.2474 (0.0030)	**0.3002*** (0.0012)	3.37%	2.42%
	N@5	0.1106 (0.0013)	0.1804 (0.0018)	0.2857 (0.0007)	0.2751 (0.0009)	0.2895 (0.0006)	0.2796 (0.0009)	0.2999 (0.0008)	0.3025 (0.0007)	0.2818 (0.0012)	0.2822 (0.0008)	0.2873 (0.0012)	0.2566 (0.0030)	**0.3099*** (0.0013)	3.33%	2.45%
	N@10	0.1478 (0.0012)	0.1945 (0.0018)	0.3044 (0.0009)	0.2928 (0.0008)	0.3079 (0.0007)	0.2981 (0.0010)	0.3172 (0.0009)	0.3197 (0.0007)	0.3002 (0.0011)	0.3007 (0.0009)	0.3059 (0.0012)	0.2762 (0.0029)	**0.3274*** (0.0013)	3.22%	2.41%

Table 3: Performance on shuffle play sessions.

Setting		Rec. SBR	SSL SBR	Graph-based SBR		Ours	Relative Gap	
Dataset	Metric	FMLP	CL4SRec	SRGNN	GCSAN	**MUSE**	Δ_b	Δ_s
MSSD 3d	R@10	0.2256 (0.0009)	0.2297 (0.0025)	0.2304 (0.0024)	0.2283 (0.0020)	**0.2401*** (0.0015)	4.21%	5.17%
	M@10	0.1071 (0.0008)	0.1080 (0.0014)	0.1140 (0.0010)	0.1137 (0.0013)	**0.1181*** (0.0008)	3.60%	3.87%
	N@10	0.1345 (0.0007)	0.1362 (0.0016)	0.1410 (0.0013)	0.1402 (0.0014)	**0.1464*** (0.0009)	3.83%	4.42%
MSSD 5d	R@10	0.2265 (0.0011)	0.2250 (0.0015)	0.2330 (0.0023)	0.2295 (0.0017)	**0.2400*** (0.0012)	3.00%	4.58%
	M@10	0.1069 (0.0010)	0.1061 (0.0008)	0.1146 (0.0010)	0.1136 (0.0010)	**0.1179*** (0.0004)	2.88%	3.79%
	N@10	0.1345 (0.0008)	0.1337 (0.0007)	0.1420 (0.0011)	0.1404 (0.0010)	**0.1462*** (0.0003)	2.96%	4.13%

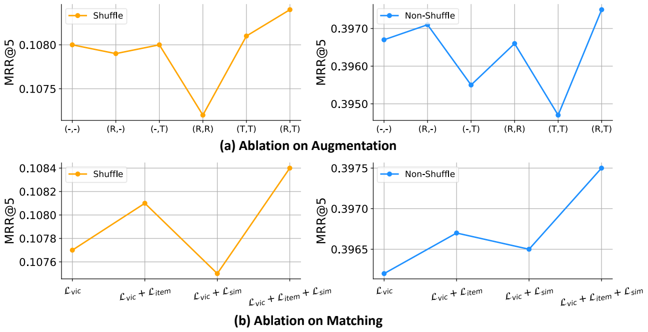
Table 4: Performance on non-shuffle play sessions.

Setting		Rec. SBR	SSL SBR	Graph-based SBR		Ours	Relative Gap	
Dataset	Metric	FMLP	CL4SRec	SRGNN	GCSAN	**MUSE**	Δ_b	Δ_s
MSSD 3d	R@10	0.4868 (0.0032)	0.4776 (0.0029)	0.4885 (0.0029)	0.4963 (0.0038)	**0.5034*** (0.0037)	3.05%	1.43%
	M@10	0.3728 (0.0026)	0.3620 (0.0032)	0.3841 (0.0034)	0.3943 (0.0024)	**0.3992*** (0.0030)	3.93%	1.24%
	N@10	0.4001 (0.0028)	0.3897 (0.0028)	0.4091 (0.0031)	0.4188 (0.0026)	**0.4242*** (0.0031)	3.69%	1.29%
MSSD 5d	R@10	0.4872 (0.0017)	0.4724 (0.0021)	0.4916 (0.0025)	0.4972 (0.0014)	**0.5051*** (0.0007)	2.75%	1.59%
	M@10	0.3751 (0.0006)	0.3662 (0.0008)	0.3899 (0.0007)	0.3963 (0.0008)	**0.4026*** (0.0008)	3.26%	1.59%
	N@10	0.4019 (0.0005)	0.3916 (0.0011)	0.4143 (0.0010)	0.4205 (0.0010)	**0.4272*** (0.0007)	3.11%	1.59%

Additionally, MUSE significantly surpasses other SSL approaches. It is shown that our SSL frameworks and fine-grained matching strategies facilitate the alignment of representations.

4.3 Fine-grained Performance Comparison

We delve into the examination of MUSE on fine-grained scenarios to deeply understand its benefits. We divide the test data into two subsets consisting of the shuffle and non-shuffle play sessions in Table 3 and Table 4, respectively. We make the following observations: 1) MUSE substantially bolsters the performance on the shuffle play sessions compared to baselines (see Table 3). It indicates that the transition-based augmentation and fine-grained matching strategies of MUSE are indeed beneficial to shuffle play sessions. It aligns with our primary objective of improving shuffle-play sessions by alleviating the unique transitions. 2) MUSE also boosts the performance on non-shuffle play sessions even though our framework focuses on shuffle play sessions (see Table 4). The SSL framework can enhance the representation of non-shuffle play sessions by utilizing reorder-based augmentation. 3) The performance gain of the state-of-the-art baseline, GCSAN, is biased towards non-shuffle play sessions (see Table 4), as it struggles to surpass its backbone, SRGNN, during shuffle play sessions (see Table 3). This indicates that self-attention falls short of capturing users' dynamic preferences in the shuffle play environment. In summary, MUSE demonstrates its efficacy in improving not only the performance of shuffle play sessions but also non-shuffle play sessions. This is achieved by addressing unique transitions through transition-based augmentation, enhancing robustness via fine-grained matching between



(a) Ablation on Augmentation

(b) Ablation on Matching

Figure 6: Ablation studies (MSSD-5d dataset, MRR@5). In (a), "(-,-)" denotes when augmentation is not made on both non-shuffle (left) and shuffle play sessions (right), "(R,-)", "(-,T)" denote reorder-based augmentation is made on the non-shuffle play sessions, and transition-based augmentation is made on shuffle play sessions, respectively. (R,T): MUSE.

augmented views, and mimicking shuffle play environment using reorder-based augmentation for non-shuffle play sessions. It highlights the versatility and effectiveness of MUSE in capturing user preferences across diverse music playback sessions.

4.4 Ablation Studies

Ablation on Transition-based Augmentation. In Figure 6 (a), our observations suggest that applying augmentations to both shuffle and non-shuffle play sessions yields optimal results. More specifically, non-shuffle play sessions benefit from reorder-based augmentation, while shuffle play sessions derive particular advantages from transition-based augmentation (i.e., (R,T) yields the best result). These findings underscore the significance of incorporating transition information for shuffle play sessions and highlight the efficacy of reordering tracks for non-shuffle play sessions.

Ablation on Fine-grained Matching. In Figure 6 (b), we present an ablation study to delve into the effectiveness of the fine-grained matching strategies. These strategies demonstrate enhancements in recommendations for both shuffle and non-shuffle play sessions. More precisely, item-based matching facilitates the alignment of the track embeddings of the identical items between two views. It enables us to cope with the shuffle and non-shuffle play session recommendations adeptly. As elaborated in Section 3.3, similarity-based matching complements item-based matching by considering the similarity of track representations. This synergistic combination verifies that the similarity-based method reinforces the item-based approach as envisioned, leading to a more refined and precise alignment process. In essence, employing both item-based and similarity-based matching mechanisms is pivotal in optimizing recommendation performance.

4.5 Sensitivity Analysis

The two key hyperparameters for MUSE are reordering hyperparameter γ, which determines the proportion of tracks to be reordered, and loss-controlling hyperparameter α (Eq. 13), which balances between the matching loss and alignment loss. As depicted in Figure 7, MUSE demonstrates robust performance, outperforming the state-of-the-art baseline, GCSAN [47] (0.2939 as shown in Table 2), in all combinations. We notice that a moderate reordering probability, γ (i.e., 0.5), of non-shuffle play sessions is advantageous. Excessive reordering (i.e., high γ) could hamper the original session's semantics, while too little reordering (i.e., low γ) might hinder the augmentation's potential for enhancing generalizability. This result aligns with our strategy, aiming to simulate the shuffle play context and enhance the model's capability to handle such scenarios effectively. Furthermore, opting for a low value of the loss-controlling hyperparameter, α (i.e., 0.2), proves to be advantageous for training MUSE. This is because $\mathcal{L}_{matching}$ encompasses both our proposed item- and similarity-based matching, leading to a relatively high scale of loss value. As a result, this choice acts effectively as a regularizer, contributing to the overall performance.

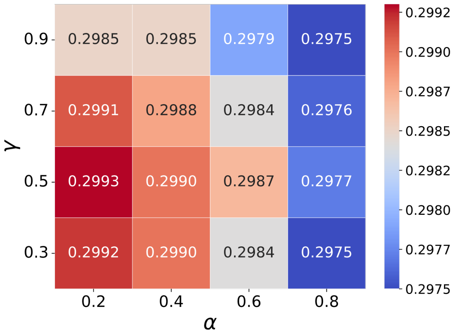


Figure 7: Sensitivity analysis (MSSD-5d dataset, MRR@5). α and γ are loss-controlling and reordering hyperparameters, respectively.

5 CONCLUSION

In this work, based on our empirical findings of the importance of shuffle play sessions in the music domain, we propose MUSE, a pioneering framework for music recommendation. Our approach employs a self-supervised learning framework to maximize the agreement between the original and augmented sessions. The augmentation is derived from a novel augmentation called transition-based augmentation, which alleviates the unique transition problem observed in shuffle play sessions by inserting the potential transition patterns. To further facilitate the alignment of representations across the two views, we introduce two precise matching strategies: the item-based approach ensuring proximity in the embedding space for identical items across both views, and the similarity-based matching strategy, which supplements the alignment of similar embeddings between the views based on the nearest neighbors of each track. Through experiments conducted across diverse environments, we demonstrate MUSE's competence, specifically in the shuffle play environment, over 12 baseline models on a large-scale Music Streaming Sessions Dataset (MSSD) from Spotify. Moreover, a detailed analysis not only confirms MUSE's effectiveness in elevating performance for shuffle play sessions but also underscores its ability to bolster outcomes in non-shuffle play environments.

Acknowledgement: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.2021R1C1C1009081) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00077).

REFERENCES

[1] Sainath Adapa. 2019. Sequential modeling of Sessions using Recurrent Neural Networks for Skip Prediction. CoRR abs/1904.10273 (2019). arXiv:1904.10273 http://arxiv.org/abs/1904.10273

[2] Rishabh Ahuja, Arun Solanki, and Anand Nayyar. 2019. Movie recommender system using k-means clustering and k-nearest neighbor. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 263–268.

[3] Khalid Anwar, Jamshed Siddiqui, and Shahab Saquib Sohail. 2020. Machine learning-based book recommender system: a survey and new perspectives. International Journal of Intelligent Information and Database Systems 13, 2-4 (2020), 231–248.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).

[5] Adrien Bardes, Jean Ponce, and Yann LeCun. 2021. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. arXiv preprint arXiv:2105.04906 (2021).

[6] Adrien Bardes, Jean Ponce, and Yann LeCun. 2022. Vicregl: Self-supervised learning of local visual features. arXiv preprint arXiv:2210.01571 (2022).

[7] Robert M Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative filtering. In KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining. sn, 7–14.

[8] Ferenc Béres. 2019. Sequential skip prediction using deep learning and ensembles.

[9] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In The World Wide Web Conference (San Francisco, CA, USA) (WWW '19). Association for Computing Machinery, New York, NY, USA, 2594–2600. https://doi.org/10.1145/3308558.3313641

[10] Manisha Chandak, Sheetal Girase, and Debajyoti Mukhopadhyay. 2015. Introducing hybrid technique for optimization of book recommender system. Procedia Computer Science 45 (2015), 23–31.

[11] Sungkyun Chang, Seungjin Lee, and Kyogu Lee. 2019. Sequential Skip Prediction with Few-shot in Streamed Music Contents. CoRR abs/1901.08203 (2019). arXiv:1901.08203 http://arxiv.org/abs/1901.08203

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In International conference on machine learning. PMLR, 1597–1607.

[13] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 15750–15758.

[14] Christina Christakou, Spyros Vrettos, and Andreas Stafylopatis. 2007. A hybrid movie recommender system based on neural networks. International Journal on Artificial Intelligence Tools 16, 05 (2007), 771–792.

[15] Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. arXiv preprint arXiv:2005.12766 (2020).

[16] Ghazal Fazelnia, Eric Simon, Ian Anderson, Benjamin Carterette, and Mounia Lalmas. 2022. Variational User Modeling with Slow and Fast Features. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (Virtual Event, AZ, USA) (WSDM '22). Association for Computing Machinery, New York, NY, USA, 271–279. https://doi.org/10.1145/3488560.3498477

[17] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. arXiv preprint arXiv:2104.08821 (2021).

[18] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems 33 (2020), 21271–21284.

[19] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Modelling Sequential Music Track Skips using a Multi-RNN Approach. CoRR abs/1903.08408 (2019). arXiv:1903.08408 http://arxiv.org/abs/1903.08408

[20] 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 (Virtual Event, Brazil) (RecSys '20). Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/3383313.3412248

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 9729–9738.

[22] Balázs Hidasi and Alexandros Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. CoRR abs/1706.03847 (2017). arXiv:1706.03847 http://arxiv.org/abs/1706.03847

[23] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015).

[24] Tomoharu Iwata, Shinji Watanabe, and Hiroshi Sawada. 2011. Fashion coordinates recommender system using photographs from fashion magazines. In Twenty-Second International Joint Conference on Artificial Intelligence.

[25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In 2018 IEEE international conference on data mining (ICDM). IEEE, 197–206.

[26] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. Advances in neural information processing systems 33 (2020), 18661–18673.

[27] Kibum Kim, Dongmin Hyun, Sukwon Yun, and Chanyoung Park. 2023. MELT: Mutual Enhancement of Long-Tailed User and Item for Sequential Recommendation. arXiv preprint arXiv:2304.08382 (2023).

[28] Manoj Kumar, DK Yadav, Ankur Singh, and Vijay Kr Gupta. 2015. A movie recommender system: Movrec. International Journal of Computer Applications 124, 3 (2015).

[29] Tuck W Leong, Frank Vetere, and Steve Howard. 2005. The serendipity shuffle. In Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future. 1–4.

[30] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 1419–1428.

[31] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015).

[32] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 1831–1839.

[33] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A simple and strong baseline for collaborative filtering. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 1243–1252.

[34] 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 and Knowledge Management (Virtual Event, Queensland, Australia) (CIKM '21). Association for Computing Machinery, New York, NY, USA, 3333–3337. https://doi.org/10.1145/3459637.3482123

[35] James R Norris. 1998. Markov chains. Number 2. Cambridge university press.

[36] Jake Olivier, William D. Johnson, and Gailen D. Marshall. 2008. The logarithmic transformation and the geometric mean in reporting experimental IgE results: what are they and when and why to use them? Annals of Allergy, Asthma & Immunology 100, 4 (2008), 333–337. https://doi.org/10.1016/S1081-1206(10)60595-9

[37] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In Proceedings of the fifteenth ACM international conference on web search and data mining. 813–823.

[38] Nikita Ramesh and Teng-Sheng Moh. 2018. Outfit recommender system. In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 903–910.

[39] Peng Shi and Edward W. Frees. 2010. Long-tail longitudinal modeling of insurance company expenses. Insurance: Mathematics and Economics 47, 3 (2010), 303–314. https://doi.org/10.1016/j.insmatheco.2010.07.005

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1 (2014), 1929–1958.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[42] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten De Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. 345–354.

[43] Robert M West. 2022. Best practice in statistics: The use of log transformation. Annals of Clinical Biochemistry 59, 3 (2022), 162–165.

[44] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, Vol. 33. 346–353.

[45] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. arXiv preprint arXiv:2012.15466 (2020).

[46] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation.

In 2022 IEEE 38th international conference on data engineering (ICDE). IEEE, 1259–1273.

[47] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation.. In IJCAI, Vol. 19. 3940–3946.

[48] Sukwon Yun, Kibum Kim, Kanghoon Yoon, and Chanyoung Park. 2022. Lte4g: long-tail experts for graph neural networks. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2434–2443.

[49] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In International Conference on Machine Learning. PMLR, 12310–12320.

[50] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In Proceedings of the 29th ACM international conference on information & knowledge management. 1893–1902.

[51] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is all you need for sequential recommendation. In Proceedings of the ACM Web Conference 2022. 2388–2399.

[52] Lin Zhu and Yihong Chen. 2019. Session-based Sequential Skip Prediction via Recurrent Neural Networks. CoRR abs/1902.04743 (2019). arXiv:1902.04743 http://arxiv.org/abs/1902.04743