Counteracting User Attention Bias in Music Streaming Recommendation via Reward Modification

Xiao Zhang*
Gaoling School of Artificial Intelligence
Renmin University of China
zhangx89@ruc.edu.cn

Sunhao Dai*
Gaoling School of Artificial Intelligence
Renmin University of China
sunhaodai@ruc.edu.cn

Jun Xu†
Gaoling School of Artificial Intelligence
Renmin University of China
junxu@ruc.edu.cn

Zhenhua Dong
Huawei Noah's Ark Lab
dongzhenhua@huawei.com

Quanyu Dai
Huawei Noah's Ark Lab
daiquanyu@huawei.com

Ji-Rong Wen
Gaoling School of Artificial Intelligence
Renmin University of China
jrwen@ruc.edu.cn

ABSTRACT

In streaming media applications, like music Apps, songs are recommended in a continuous way in users' daily life. The recommended songs are played automatically although users may not pay any attention to them, posing a challenge of *user attention bias* in training recommendation models, i.e., the training instances contain a large number of false-positive labels (users' feedback). Existing approaches either directly use the auto-feedbacks or heuristically delete the potential false-positive labels. Both of the approaches lead to biased results because the false-positive labels cause the shift of training data distribution, hurting the accuracy of the recommendation models. In this paper, we propose a learning-based counterfactual approach to adjusting the user auto-feedbacks and learning the recommendation models using Neural Dueling Bandit algorithm, called NDB. Specifically, NDB maintains two neural networks: a user attention network for computing the importance weights that are used for modifying the original rewards, and another random network trained with dueling bandit for conducting online recommendations based on the modified rewards. Theoretical analysis showed that the modified rewards are statistically unbiased, and the learned bandit policy enjoys a sub-linear regret bound. Experimental results demonstrated that NDB can significantly outperform the state-of-the-art baselines.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Reinforcement learning**.

*Equal contribution.
†Jun Xu is the corresponding author. The work was partially done at Beijing Key Laboratory of Big Data Management and Analysis Methods.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539393

KEYWORDS

user attention bias, dueling bandit, streaming recommendation

ACM Reference Format:
Xiao Zhang, Sunhao Dai, Jun Xu, Zhenhua Dong, Quanyu Dai, and Ji-Rong Wen. 2022. Counteracting User Attention Bias in Music Streaming Recommendation via Reward Modification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539393

1 INTRODUCTION

With the increasing popularity of on-demand streaming services, music streaming recommendation has attracted increasing research attentions in recent years [4, 8, 36]. Typically, a music App would provide the users playlists according to their music preferences. The songs in the playlists will be automatically played until the users explicitly stop/skip the songs, or reach the end of the playlists. Therefore, one important task is how to automatically generate playlists that well fit the music preferences of users [6, 33].

One approach to music streaming recommendation is treating the problem of automatic playlist generation as a process of sequential song selection. As shown in Figure 1, after recommending the first song to a user, the next song is selected from the candidate songs based on the user's contexts, and this process is repeated until the user exits the playlist/App. During the sequential selection process, the model is updated according to the user feedbacks in an online manner. Thus, it is critical for a music recommendation model to receive and leverage accurate user feedbacks.

After a user receives a recommended song, she may play the entire song (positive feedback) or skip the song before the ending of this song (negative feedback). Unfortunately, some of the positive feedbacks are false since the songs in a playlist are usually consecutively streamed and played automatically without any user behaviors on the system. Specifically, songs are typically played in the background while users may pay less attention to the music App and perform other activities such as exercising, late-night reading, or whilst commuting [15, 35].

This phenomenon has been empirically demonstrated in Figure 2: (a) the frequency of users' active actions (i.e., skip actions) gradually decrease indicating that the user's attention is gradually losing over

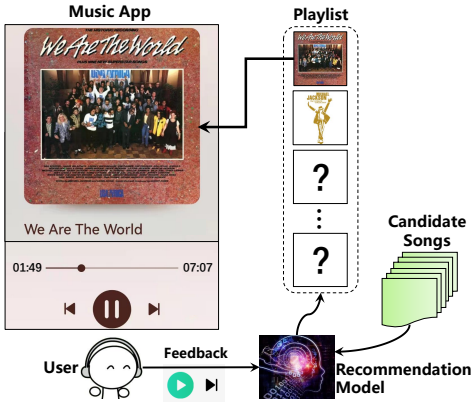

Figure 1: The task of automatically generation of playlists in music streaming recommendation scenario.

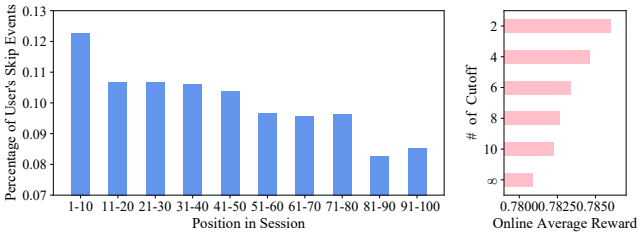

Figure 2: Empirical study of user attention on the data on Last.fm [3]. Left: Percentage of user's skip events w.r.t. the position of the recommended songs in one session. Right: Recommendation performance (online average reward defined in Section 6.1.3) of SBUCB [16] equipped with a play cutoff technology [25], where only the first C songs played entirely after the user's skip action were considered as the positive user feedbacks and the rest of user feedbacks were ignored. C is called the '# of Cutoff' and '∞' means that all the observed user feedbacks were used for training.

time; (b) Simply discarding some of the positive user feedbacks in training can help improve the performance of recommendation models. In other words, when a user is performing activities other than listening to music, a song is played automatically even if the song does not fit her music preference. As a result, some songs that are not liked by the user are played completely, that is, some positive user feedbacks are then falsely labeled and stored in the user's historical listening activities (*false-positive labels* exist). The feedback distribution of the training instances is different from the user's real preferences, which ultimately hurts the recommendation performances. In this paper, we call this category of biases in user feedback due to the lack of concentration as the *user attention bias*.

In order to counteract the user attention bias, it is necessary to model the mechanism of both the user attention and the user's musical preferences simultaneously [1]. Existing music streaming recommendation approaches, however, either directly learn the recommendation model from the user's listening feedback [4, 21], or use heuristic algorithms to address user attention bias [25] overlooking the contextual information of both the user and songs. Little research efforts have been paid on modeling user attention in streaming recommendation scenarios where users sometimes listen to music in a passive mode and a large number of false-positive labels exist in the log data.

In this paper, we propose a counterfactual learning approach to adjusting the falsely labeled positive feedbacks of users and learning users' musical preferences using a dueling bandit model, referred to as NDB. To achieve unbiased estimates of the real user feedbacks, NDB maintains a recurrent neural network to capture the mechanism of user attention on the recommender system, and learns a randomly weighted neural network using random Fourier features for predicting the relevance. In this way, the importance weights can be derived from the predicted probability distributions of the user attention and relevance, and used for obtaining the unbiased modified rewards. Then, the relevance prediction model can be updated based on the modified rewards via dueling bandit, and adopted as a recommendation model for conducting music streaming recommendations. Theoretical analysis demonstrated that the reweighted rewards are statistically unbiased of the true rewards, and the bandit policy enjoys a sub-linear regret bound.

2 RELATED WORK

User modeling in music recommendation aims to analyze and simulate the user behaviors in music services, which can help improve the performance of music recommendation models [29, 34, 39]. Cheng et al. [10] presented a personalized dual-layer topic model for the music retrieval task, which captures users' music preference on songs via the connection of latent semantic spaces. Ferwerda and Schedl [12] proposed a personality-based music recommendation model by modeling users' personality traits, in which the relationships between users' personality and their behavior, preferences, and needs are identified. Reza Aditya Permadi [25] designed a heuristic cut-off technology to address user attention bias, which simply treats some part of music videos watched completely after an active user action on the system as positive user feedbacks. Since behavior patterns are usually different among users on different contents, contexts of both the users and songs are crucial for accurate recommendations, and learning-based debiasing approaches for user attention bias need to be further studied.

Bandit-based streaming recommendation has received considerable attention over the past several years, which aims to maximize the cumulative reward feedback from users by training and applying the recommendation model in an online manner. There are several types of bandit approaches which are commonly used in streaming recommender systems, including contextual bandit [4, 19], batched bandit [16, 40], bandit optimization [21, 27, 38]. Bendada et al. [4] proposed a contextual bandit model for music playlist recommendation with multiple user plays, which leverages swipeable carousels to recommend personalized content to their users. Zhang et al. [40] presented a batched bandit approach for streaming recommendation with delayed feedback, where a survival model is maintained to capture the delay mechanism and the online recommendation model is fixed. Pereira et al. [21] introduced an efficient bandit optimization algorithm for music streaming recommendation using dueling bandit, which updates the recommendation model using negative user feedbacks. Although bandits have been

Table 1: A summary of notations.

Symbol	Explanation
$[m]$	$\{1, 2, \ldots, m\}$
N	Number of episodes
B_n	Batch size in the n-th user session (i.e., the n-th episode)
$\boldsymbol{u}_n \in \mathbb{R}^{d_u}$	Feature vector of the user in the n-th user session
$\boldsymbol{s} \in \mathbb{R}^{d_s}$	Feature vector of a candidate song
$\boldsymbol{x}_n(\boldsymbol{s})$	Context vector corresponding to the song s and the user $\boldsymbol{u}_n$ which is represented by $\boldsymbol{x}_n(\boldsymbol{s}) = \left[\boldsymbol{u}_n^\intercal, \boldsymbol{s}^\intercal\right]^\intercal$
$c \in \{0, 1\}$	Observed user feedback indicating whether the user skips the song ($c = 0$) or plays the entire song ($c = 1$)
$v \in \{0, 1\}$	True user preference (i.e., relevance) indicating whether the user prefers recommended song (may be unobserved)
$o \in \{0, 1\}$	Attention variable indicating whether the user pays attention to the recommended song (may be unobserved)

extensively studied in streaming recommendation, how to deal with user attention bias in an online manner is still an unsolved problem.

3 PROBLEM FORMULATION

This section formulates the problem of music streaming recommendation (MSR) with bandit and analyzes the user attention bias in MSR.

3.1 Music Streaming Recommendation (MSR)

MSR can be formulated as a problem of sequential decision making where each user consumes a music streaming service through a music-listening session. More specifically, during a music-listening session of each user (i.e., one episode), the candidate song set can be viewed as the action space $\mathcal{S}$ of an episodic bandit. Then, given a context space $\mathcal{X}$ that summarizes the information of both the user and candidate songs, a song is selected by a recommendation policy π and sequentially recommended to the user. After one song is recommended, the song may be entirely played by the user (i.e, positive user feedback) or skipped before its ending (i.e, negative user feedback). Finally, the bandit reward r defined on the user feedback is received and can be used for updating the current recommendation policy. The updated policy, thereafter, will be adopted for recommending the next song.

The above process can be formalized as an *episodic bandit*, and represented using a 5-tuple $\langle N, \mathcal{S}, \mathcal{X}, \pi, r\rangle$:

Number of episodes N. The sequential decision process includes N episodes, where each episode corresponds to a user session. In the n-th user session (i.e., the n-th episode), the agent interacts with one user B_n times (i.e., recommends B_n songs to the user in an online manner). B_n is called the **batch size** in the n-th user session.

Action space $\mathcal{S}$ denotes a given candidate action set, where each action (also called arm) corresponds to a specified candidate song. Each song in $\mathcal{S}$ is represented as a feature vector $\boldsymbol{s} \in \mathbb{R}^{d_s}$, where d_s is the feature dimension. At each step in one episode, a dynamic action space is selected as the candidate song set for recommendation. That is, at the step b in the n-th user session, a dynamic action space $\mathcal{S}_{n,b} \subseteq \mathcal{S}$ is recalled by some strategy, and choosing an action $\boldsymbol{s}_{I_{n,b}}$ from $\mathcal{S}_{n,b}$ means that the corresponding song is recommended to the user, where $I_{n,b}$ denotes the index of the selected song in the candidate song set $\mathcal{S}_{n,b}$.

Context space $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ denotes a context space that summarizes feature information of both the users and songs, where d_x is the dimension of contexts. In recommendation, a user is typically expressed as a feature vector $\boldsymbol{u} \in \mathbb{R}^{d_u}$. In particular, the user in the n-th user session is represented by a feature vector $\boldsymbol{u}_n \in \mathbb{R}^{d_u}$. Then, at step b in the n-th user session, for a candidate song $\boldsymbol{s} \in \mathcal{S}_{n,b}$, the corresponding context $\boldsymbol{x}_n(\boldsymbol{s})$ can be represented by the concatenation of the user features $\boldsymbol{u}_n$ and the song features $\boldsymbol{s}$, i.e., $\boldsymbol{x}_n(\boldsymbol{s}) = [\boldsymbol{u}_n^\intercal, \boldsymbol{s}^\intercal]^\intercal \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ where $d_x = d_u + d_s$.

Policy $\pi : \mathcal{X} \to \mathcal{S}$ describes the decision-making rule of an agent (i.e., the recommendation model), which selects action for execution according to the relevance score of each action. At step b in the n-th user session, given a set of candidate songs $\mathcal{S}_{n,b}$, a relevance score function (or relevance probability) f parameterized by $\boldsymbol{\theta}_{n,b}$ treats the context $\boldsymbol{x}_n(\boldsymbol{s}) \in \mathcal{X}$ as inputs and determines which action to take: $\boldsymbol{s}_{I_{n,b}} := \arg\max_{\boldsymbol{s} \in \mathcal{S}_{n,b}} f(\boldsymbol{x}_n(\boldsymbol{s})|\boldsymbol{\theta}_{n,b})$.

Reward r is defined upon the user feedback. Specifically, after recommending a song $\boldsymbol{s} \in \mathcal{S}$ to a user $\boldsymbol{u}$, a corresponding user feedback $c \in \{0, 1\}$ is observed, which implicitly indicates whether the user feedback is negative ($c = 0$, skip before the ending of this song) or positive ($c = 1$, play the entire song). Then the observed reward $r = 1$ if $c = 1$ and $r = 0$ otherwise. Let's use the unobservable variable $v \in \{0, 1\}$ to indicate whether a user prefers the recommended song over other candidate songs, i.e., the relevance. Unfortunately, the user feedback c may not be consistent with the true but unobservable relevance $v \in \{0, 1\}$ if the user did not pay attention to the recommended song $\boldsymbol{s}$.

Table 1 summarizes the notations used throughout this paper.

3.2 User Attention Bias in MSR

In this section, we formally describe the user attention bias defined in the introduction. As defined in Table 1, we introduce the attention variable $o \in \{0, 1\}$ for indicating user's attention while listening to music, where $o = 1$ if user pays attention to the recommended song and $o = 0$ otherwise. Ideally, after receiving a recommended song, a user will *play* the entire song if she prefers this song, or *skip* the song if the song does not fit her musical tastes. In other words, the observed user feedback $c \in \{0, 1\}$ should be consistent with the true relevance $v \in \{0, 1\}$, i.e., given a context $\boldsymbol{x}_n(\boldsymbol{s}) \in \mathcal{X}$ in the n-th user session,

$$\Pr\{c = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\} = \Pr\{v = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\}, \tag{1}$$

where $\Pr\{v = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\}$ denotes the *relevance probability* indicating how much the user $\boldsymbol{u}_n$ prefers the recommended song $\boldsymbol{s}$. However, as summarized in Table 2, the positive user feedbacks will be false if the user loses her attention to the music app and songs are played automatically that do not fit her music preference. Then, different from the ideal user model in Eq.(1), an attention-based user model can be derived from Table 2:

$$\begin{aligned} &\Pr\{c = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\} \\ &= \Pr\{o = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\} \cdot \Pr\{v = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\} + \Pr\{o = 0|\, \boldsymbol{x}_n(\boldsymbol{s})\}, \end{aligned} \tag{2}$$

where $\Pr\{o = 1|\, \boldsymbol{x}_n(\boldsymbol{s})\}$ is called the *attention probability* that represents the probability that user attention is on the recommender system. From Eq.(2) we can observe that the probability distribution of observed user feedback c depends on not only the true relevance v but also the attention variable o, and the user attention bias is caused by the loss of user attention. Thus, to counteract the user

Table 2: Relation among the variables in presence of user attention bias. 'Attention': attention variable; 'Relevance': true relevance; 'True/False': variable indicating whether the observed user feedback c is true or false.

Attention o	Relevance v	User Feedback c	True/False
1	1	1 (play)	True
1	0	0 (skip)	True
0	1	1 (play)	True
0	0	1 (play)	**False**

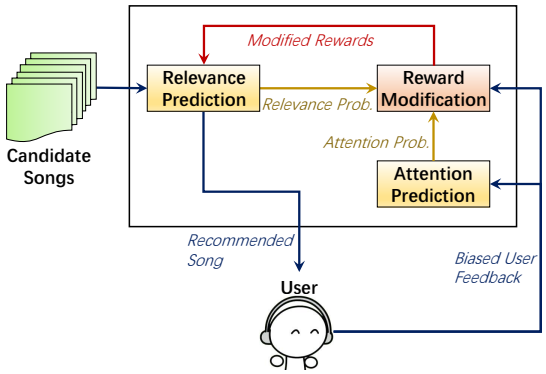

Figure 3: The overall architecture of the proposed NDB.

attention bias, it is essential to maintain learning-based models for both the user attention and her music preference.

4 NDB: THE PROPOSED APPROACH

In this section, we present a novel Neural Dueling Bandit approach named NDB which modifies the observed rewards using counterfactual learning for counteracting the user attention bias.

4.1 Approach Overview

Figure 3 illustrates the overall architecture of the proposed NDB approach. NDB consists of three ingredients: (1) *reward modification* tries to counteract the attention bias in user feedbacks and construct the unbiased modified rewards through the prediction of the probabilities of the user attention and the relevance; (2) *attention prediction* predicts the attention probability of each user on the recommended songs, which is implemented using recurrent neural networks; (3) *relevance prediction* aims to predict the user preference for the candidate songs and choose the most relevant song for online recommendation, where the prediction model is formulated using randomly weighted neural network that can be trained online based on the modified rewards using dueling bandit.

Next, we specify the ingredients of NDB with details.

4.2 Reward Modification using Importance Sampling

In the n-th user session, after recommending a song $\boldsymbol{s}$ to a user $\boldsymbol{u}_n$, her feedback $c \in \{0, 1\}$ is received. Then, we can specify the *observed reward* by $r(\boldsymbol{x}_n(\boldsymbol{s}), c) := c$, where $\boldsymbol{x}_n(\boldsymbol{s}) = [\boldsymbol{u}_n^\intercal, \boldsymbol{s}^\intercal]^\intercal \in \mathcal{X}$ is the context. But the *true reward* is determined by the true relevance v and denoted by $r(\boldsymbol{x}_n(\boldsymbol{s}), v) := v$, and the goal of recommendation should be maximizing the following *expected true reward*:

$$\mathbb{E}_v \left[r(\boldsymbol{x}_n(\boldsymbol{s}), v) \right] = \Pr \left\{ v = 1 | \ \boldsymbol{x}_n(\boldsymbol{s}) \right\}.$$

In comparison with the expected true reward, the *expected observed reward* can be expressed by:

$$\mathbb{E}_c \left[r(\boldsymbol{x}_n(\boldsymbol{s}), c) \right] = \Pr \left\{ c = 1 | \ \boldsymbol{x}_n(\boldsymbol{s}) \right\}.$$

From Eq.(2) we have $\Pr \{v = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\} \neq \Pr \{c = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\}$ usually holds, and thus we can conclude that the observed reward is biased for estimating the expected true reward. To counteract the biases in observed rewards caused by the loss of user attention, we propose to modify the biased observed rewards using counterfactual learning. Specifically, by leveraging importance sampling approach [7, 30, 37, 40], the following *modified rewards* can be obtained:

$$r^{\text{mod}}(\boldsymbol{x}_n(\boldsymbol{s}), c) := w \cdot r(\boldsymbol{x}_n(\boldsymbol{s}), c), \tag{3}$$

where w denotes the *importance weight* that is defined as follows:

$$\begin{aligned} w &:= \frac{\Pr \{v = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\}}{\Pr \{c = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\}} \\ &= 1 \Big/ \left[\Pr \{o = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\} + \frac{\Pr \{o = 0 | \ \boldsymbol{x}_n(\boldsymbol{s})\}}{\Pr \{v = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\}} \right]. \end{aligned} \tag{4}$$

The last equality in Eq.(4) is obtained by substituting the proposed attention-based user model Eq.(2). Theoretically, we can easily prove the unbiasedness of the obtained modified rewards equipped with the importance weights in Eq.(4) for estimating the expected true reward, shown in Theorem 4.1.

Theorem 4.1 (Unbiasedness of Modified Rewards). *In the n-th user session, for any context $\boldsymbol{x}_n(\boldsymbol{s}) \in \mathcal{X}$, the modified reward using the importance weight in Eq.(4) is an unbiased estimate of the expected true reward, i.e., $\mathbb{E}_c[r^{\text{mod}}(\boldsymbol{x}_n(\boldsymbol{s}), c)] = \mathbb{E}_v \left[r(\boldsymbol{x}_n(\boldsymbol{s}), v) \right]$.*

From the formulation in Eq.(4), we observe that the importance weights satisfy $w \leq 1$, and a larger probability $\Pr \{o = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\}$ leads to a larger weight closed 1. The intuition is that the attention probability $\Pr \{o = 1 | \ \boldsymbol{x}_n(\boldsymbol{s})\}$ can be interpreted as a confidence score indicating if the positive user feedback $c = 1$ is true, and a more reliable observed reward deserves a larger weight. Another observation from Eq.(4) is that, to estimate the importance weight w, the attention probability and the relevance probability need to be predicted simultaneously, and the prediction components will be introduced in the following sections.

4.3 Attention Prediction with GRUs

To predict the attention probability $\Pr(o = 1 | \boldsymbol{x}_n(\boldsymbol{s}))$ in Eq.(4), we propose a user attention model g using recurrent neural networks, which is parameterized by $\boldsymbol{\beta}_n$ in the n-th user session. The attention model g enables NDB to take the contextual information (i.e., the user listening history and her feedbacks in this session) into consideration. At step b in the n-th user session, we first collect the recommended songs and user feedbacks that have so far received in this session and store them into a buffer $\mathcal{D}_{n,b} := \{(u_n, \boldsymbol{s}_{I_{n,j}}, c_{n,j})\}_{j \in [b]}$, where $c_{n,j}$ denotes the user feedback received at step j in this session. Then, the *predicted attention probability* at step b in the n-th user session, denoted by $p_{n,b}$, can be obtained by:

$$p_{n,b} := g(\mathcal{D}_{n,b} \mid \boldsymbol{\beta}_n), \tag{5}$$

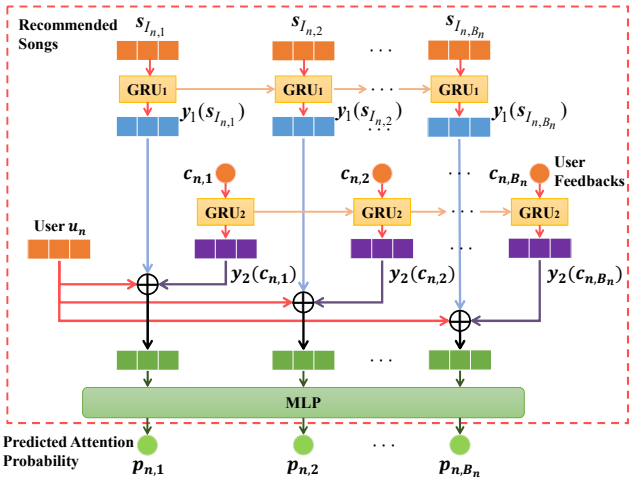

Figure 4: The component of attention prediction with GRUs.

where the obtained $p_{n,b}$ is an estimate of the true attention probability $\Pr\{o = 1 | \boldsymbol{x}_n(\boldsymbol{s}_{I_{n,b}})\}$. As illustrated in Figure 4, we implement the user attention model g using one MLP (Multi-Layer Perceptron) and two GRUs (Gated Recurrent Units) [11]. Specifically, we formulate the attention model g in Eq.(5) as follows:

$$g(\mathcal{D}_{n,b} \mid \boldsymbol{\beta}_n) = \sigma(\text{MLP}(\boldsymbol{y}_1(\boldsymbol{s}_{I_{n,b}}) \oplus \boldsymbol{y}_2(c_{n,b}) \oplus \boldsymbol{u}_n)),$$

where '$\oplus$' is an operation to concatenate two vectors together, and $\sigma(\cdot)$ is the element-wise sigmoid function, and the representations $y_1(\boldsymbol{s}_{I_{n,b}})$ and $\boldsymbol{y}_2(c_{n,b})$ are obtained by two GRUs. Formally, in each session, (a) GRU_1 scans the historical listening songs as follows: at step b, it takes the embedding of the recommended song $\boldsymbol{s}_{I_{n,b}}$ as input, and output the representation $\boldsymbol{y}_1(\boldsymbol{s}_{I_{n,b}})$:

$$\boldsymbol{y}_1(\boldsymbol{s}_{I_{n,b}}), \boldsymbol{h}_{n,b} = \text{GRU}_1(\boldsymbol{s}_{I_{n,b}}, \boldsymbol{h}_{n,b-1}),$$

where $\boldsymbol{h}_{n,b}$ and $\boldsymbol{h}_{n,b-1}$ are the hidden vectors at the b-th and $(b-1)$-th steps, respectively; (b) another GRU_2 is used to process the user feedback sequence as follows: $\boldsymbol{y}_2(c_{n,b}), \boldsymbol{h}'_{n,b} = \text{GRU}_2(c_{n,b}, \boldsymbol{h}'_{n,b-1})$, where the output $\boldsymbol{y}_2(c_{n,b})$ is the representation of the user feedback at step b in the n-th user session, and $\boldsymbol{h}'_{n,j}$ denotes the hidden vector in GRU_2 at the j-th step. The estimate of $p_{n,b}$ can also be achieved using randomized experiments or unbiased methods such as [2].

4.4 Relevance Prediction using RWNN

For predicting the relevance probability $\Pr(v = 1 | \boldsymbol{x}_n(S))$ in Eq.(4), we adopt a Randomly Weighted Neural Network (RWNN) that can be viewed as a single-hidden-layer neural network with randomly drawn weights as a encoder [24, 31]. As an empirical implementation of RWNN, we use the random Fourier features [23] as the random hidden layer. Specifically, at step b in the n-th user session, for a candidate song $\boldsymbol{s} \in \mathcal{S}_{n,b}$, its relevance probability can be predicted by the RWNN $f(\cdot|\boldsymbol{\theta}_{n,b})$ as follows:

$$f(\boldsymbol{x}_n(\boldsymbol{s})|\boldsymbol{\theta}_{n,b}) := \sigma\left(\boldsymbol{\theta}_{n,b}^{\intercal} \phi(\boldsymbol{x}_n(\boldsymbol{s}))\right), \ \phi(\boldsymbol{x}_n(\boldsymbol{s})) := \frac{\cos\left(G\boldsymbol{x}_n(\boldsymbol{s}) + \boldsymbol{\rho}\right)}{\sqrt{d/2}},$$

where $\boldsymbol{\theta}_{n,b} \in \mathbb{R}^d$ denotes the model parameters of f, d is the dimension of the model parameters, $G \in \mathbb{R}^{d \times d_x}$ is a random matrix with each entry drawn independently according to the Gaussian distribution $\mathcal{N}(0, \xi^2)$, $\boldsymbol{\rho} \in \mathbb{R}^d$ is a random vector drawn i.i.d. from $[-\pi, \pi]$ uniformly, $\cos(\cdot)$ is an element-wise cosine function, and $\sigma(\cdot)$ is a sigmoid function. An intuitive interpretation of the random mapping $\phi(\cdot)$ is that the contexts are projected onto a novel feature space that is an unbiased estimate of the Gaussian reproducing kernel Hilbert space [23, 41], which can help capture the nonlinear relation between the context and the relevance.

Finally, the relevance probability and the attention probability can be predicted by $f(\boldsymbol{x}_n(\boldsymbol{s})|\boldsymbol{\theta}_{n,b})$ and $p_{n,b}$ in Eq.(5), respectively. Then, from Eq.(3) and Eq.(4), the modified reward at step b in the n-th user session can be expressed by:

$$r_{n,b}^{\text{mod}} := r_{n,b} \Big/ \left[p_{n,b} + (1 - p_{n,b}) / f(\boldsymbol{x}_n(\boldsymbol{s}_{I_{n,b}})|\boldsymbol{\theta}_{n,b})\right],$$

where $r_{n,b} := r(\boldsymbol{x}_n(\boldsymbol{s}_{I_{n,b}}), c_{n,b}) = c_{n,b}$ denotes the observed reward for the recommended song $\boldsymbol{s}_{I_{n,b}}$.

4.5 Model Training and Online Recommendation

4.5.1 Offline Training. The model parameter $\boldsymbol{\beta}$ of the user attention model g is updated at the end of each user session, which can be viewed as a offline training process after collecting the user listening history and the user feedbacks at one user session. When conducting the offline training for the neural networks in g, a binary cross entropy loss is adopted as the objective function that is solved by Adam optimizer. Since the attention variable o is unobservable, imputed attention variables are used to compute the loss function. Specifically, if a user skips a song (i.e., $c = 0$), we consider that the corresponding attention variable should be $o = 1$; if a user plays more than ten songs without any active user action (i.e., skip), the corresponding attention variable o is considered as 0 until the next active user action occurs. Although some of the negative samples (i.e., $o = 0$) may be mislabeled, it is sufficient to provide useful supervised signals for the training of the attention model.

4.5.2 Online Training. To capture the user's preference shifts online, we update the relevance prediction model f using dueling bandit [27, 38] in an online manner. More specifically, at step b in the n-th user session, given the model parameters $\boldsymbol{\theta}_{n,b}$ of f, we first choose a dueling model parameterized by $\tilde{\boldsymbol{\theta}}_{n,b}$ through performing $\tilde{\boldsymbol{\theta}}_{n,b} = \boldsymbol{\theta}_{n,b} + \delta \boldsymbol{q}$, where $\boldsymbol{q} \in \mathbb{R}^d$ is a random vector i.i.d. drawn from a uniform distribution $\mathbb{U}_s$ over the unit sphere in Euclidean space, and $\delta > 0$ is an exploration parameter. To evaluate the dueling model, instead of interacting with users using the dueling model directly, we compute the imputed reward of the dueling model $\tilde{\boldsymbol{\theta}}_{n,b}$ by $\tilde{r} := \tilde{r}(\tilde{\boldsymbol{\theta}}_{n,b}) = 1/\text{rank}_{\tilde{\boldsymbol{\theta}}_{n,b}}(\boldsymbol{s}_{I_{n,b}})$, where the function $\text{rank}_{\boldsymbol{\theta}}(\boldsymbol{s})$ outputs the position of $\boldsymbol{s}$ in the ranking list in which the candidate songs in $\mathcal{S}_{n,b}$ are sorted by the descending order of the predicted relevance probability $f(\boldsymbol{x}_n(\cdot)|\boldsymbol{\theta})$. Then, if the imputed reward $\tilde{r}$ of the dueling model satisfies $\tilde{r} > r_{n,b}^{\text{mod}}$ (i.e., the dueling model is the winner with a larger reward), $\tilde{r}$ will be used for updating the parameters of $f(\cdot|\boldsymbol{\theta}_{n,b})$ as follows:

$$\boldsymbol{\theta}_{n,b} = \boldsymbol{\theta}_{n,b} - \gamma \left(\tilde{r} - r_{n,b}^{\text{mod}}\right) \boldsymbol{q}, \tag{6}$$

where $\gamma > 0$ is the stepsize, and $\boldsymbol{q}$ is the random vector as mentioned before. Intuitively, in Eq.(6), a greater winning of a dueling model (i.e., a larger $\tilde{r}$) leads to a larger distance between the current

parameters and the updated parameters. Theoretically, the updating in Eq.(6) is equivalent to minimizing a given loss defined in Section 5 that measures the difference between $\tilde{r}$ and $r_{n,b}^{\text{mod}}$, where $(\tilde{r} - r_{n,b}^{\text{mod}})\boldsymbol{q}$ is an effective gradient estimate of the given loss.

To reduce the variance in Eq.(6), we perform the above updating process K times (i.e., K dueling models are explored and used for training parameters $\boldsymbol{\theta}_{n,b}$).

4.5.3 Online Recommendation. At step b in the n-the user session, we select a song from the candidate songs $\mathcal{S}_{n,b}$ according to the predicted relevance: $\boldsymbol{s}_{I_{n,b}} = \arg\max_{\boldsymbol{s}\in\mathcal{S}_{n,b}} f(\boldsymbol{x}_n(\boldsymbol{s})|\boldsymbol{\theta}_{n,b})$, and recommend it to the user $\boldsymbol{u}_n$. Finally, we summarize the above steps of model training and online recommendation in Algorithm 1, called NDB. We can observe that, the time complexity of online training and online recommendation in NDB is of order $O(KdM + d_x dM)$ at each step, where M denotes the maximal number of candidate songs at one step, d the dimension of model parameters in f, d_x the dimension of contexts, and K the number of dueling models.

5 DISCUSSION

Regret analysis. When the modified reward r^{mod} is close to the imputed reward $\tilde{r}$, the model parameters $\boldsymbol{\theta}$ in our recommendation policy can converge to the optimal model parameters. Letting $\Omega \in \mathbb{R}^d$ be the model parameter space of the relevance prediction model f, we introduce the loss function $\ell_{n,b}(\boldsymbol{\theta}) := \max\{0, \tilde{r}(\boldsymbol{\theta}) - r_{n,b}^{\text{mod}}\}, \boldsymbol{\theta} \in \Omega$, to measure the difference between the rewards. To evaluate the convergence of the proposed neural dueling bandit, assuming that the batch size $B_n = B, \forall n \in [N]$, we define the *regret* as follows:

$$\text{Reg}(\{\boldsymbol{\theta}_{n,b}\}_{n\in[N],b\in[B]}, \boldsymbol{\theta}^*) := \sum_{n\in[N],b\in[B]} \left[\ell_{n,b}(\boldsymbol{\theta}_{n,b}) - \ell_{n,b}(\boldsymbol{\theta}^*)\right],$$

where $\boldsymbol{\theta}^* \in \Omega$ denotes the optimal model parameters satisfying $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Omega} \sum_{n\in[N],b\in[B]} \ell_{n,b}(\boldsymbol{\theta})$. Next, we prove the regret upper bound of NDB (Algorithm 1) as follows.

Theorem 5.1 (Regret Upper Bound). *Let $T = B \times N$ be the overall number of interaction steps. For any $\boldsymbol{\theta} \in \Omega$, assume that $\tilde{r}(\boldsymbol{\theta})$ is Lipschitz continuous with Lipschitz constant L (not necessarily convex or differentiable) and $\|\boldsymbol{\theta}\|_2^2 \le C_{\boldsymbol{\theta}}$. Let $\mathbb{U}_{\text{b}}$ be a uniform distribution over the Euclidean unit ball, and $\hat{\ell}_{n,b}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{u}\in\mathbb{U}_{\text{b}}}[\ell_{n,b}(\boldsymbol{\theta} + \delta\boldsymbol{u})]$ be a differentiable version of $\ell_{n,b}$ that has Lipschitz continuous gradients with constant $L_{\text{g}} \le \delta$. Then, for an arbitrary sequence of contexts,*

$$\mathbb{E}\left[\text{Reg}(\{\boldsymbol{\theta}_{n,b}\}_{n\in[N],b\in[B]}, \boldsymbol{\theta}^*)\right] \le \frac{d}{2\gamma\delta} C_{\boldsymbol{\theta}} + \frac{d\gamma}{\delta} T + 2(C_{\boldsymbol{\theta}} + L)\delta T.$$

Remark 1. *Setting $\gamma = \sqrt{C_{\boldsymbol{\theta}}/(2T)}, \delta = d^{1/2}(C_{\boldsymbol{\theta}}/2)^{1/4}(2C_{\boldsymbol{\theta}} + 3L)^{-1/2}T^{-1/4}$, we obtain that the regret bound is of order $O(d^{1/2}T^{3/4})$, which matches the optimal regret bounds of the classic dueling bandits for the case that the user attention bias does not exist [28, 38]. The sub-linear regret bound of NDB indicates the effectiveness of approximating the optimal policy in streaming recommendation scenarios.*

Difference with dueling bandits. Compared to existing dueling bandit approaches for streaming recommendation [21, 27, 38], NDB has the following striking differences: (1) Instead of directly applying the observed rewards that may be biased, NDB uses the modified and unbiased rewards for online training; (2) NDB adopts neural networks for accurate prediction rather than a simple (generalized) linear model; (3) the parameter updating process in Eq.(6) is derived from a potential loss function $\ell_{n,b}$, enjoying a sub-linear regret guarantee even without the assumption of convex losses.

Algorithm 1: Neural Dueling Bandit (NDB)

INPUT: Batch sizes $\{B_n\}_{n\in[N]}$, number of episodes N, stepsize γ, exploration parameter δ, number of dueling models K

Initialize $\boldsymbol{\beta}_1$ with Xavier Normal, $\boldsymbol{\theta}_{1,1} \sim \mathcal{N}(0,1)$, and $\mathcal{D}_{1,0} = \emptyset$
for $n = 1$ **to** N **do**
  // Online Recommendation and Online Training
  Receive a user feature vector $\boldsymbol{u}_n$
  for $b = 1$ **to** B_n **do**
    Observe the set of candidate songs $\mathcal{S}_{n,b}$
    Obtain the contexts $\boldsymbol{x}_n(\boldsymbol{s}) \leftarrow [\boldsymbol{u}_n^\intercal, \boldsymbol{s}^\intercal]^\intercal, \forall \boldsymbol{s} \in \mathcal{S}_{n,b}$
    Recommend song $\boldsymbol{s}_{I_{n,b}} \in \mathcal{S}_{n,b}$ to the user following $\boldsymbol{s}_{I_{n,b}} \leftarrow \arg\max_{\boldsymbol{s}\in\mathcal{S}_{n,b}} f(\boldsymbol{x}_n(\boldsymbol{s})|\boldsymbol{\theta}_{n,b})$
    Receive feedback $c_{n,b}$ ($c_{n,b} = 0$ for skip and 1 for play)
    $\mathcal{D}_{n,b} \leftarrow \mathcal{D}_{n,b-1} \cup \{(u_n, \boldsymbol{s}_{I_{n,b}}, c_{n,b})\}$
    if $c_{n,b} = 1$ **then**
      Set reward $r_{n,b} \leftarrow 1$
      Compute attention probability $p_{n,b} \leftarrow g(\mathcal{D}_{n,b} \mid \boldsymbol{\beta}_n)$
      Estimate $w \leftarrow 1/[p_{n,b} + (1 - p_{n,b})/f(\boldsymbol{x}_n(\boldsymbol{s}_{I_{n,b}})|\boldsymbol{\theta}_{n,b})]$
      Reweight reward $r_{n,b}^{\text{mod}} \leftarrow w \cdot r_{n,b}$
    else
      Compute reward $r_{n,b}^{\text{mod}} \leftarrow 0$
    end if
    for $k = 1$ **to** K **do**
      Pick $\boldsymbol{q} \sim \mathbb{U}_{\text{s}}$ and update $\tilde{\boldsymbol{\theta}}_{n,b} \leftarrow \boldsymbol{\theta}_{n,b} + \delta\boldsymbol{q}$
      Compute the imputed reward of recommending song $\boldsymbol{s}_{I_{n,b}}$ using $\tilde{\boldsymbol{\theta}}_{n,b}$ as $\tilde{r} \leftarrow \tilde{r}(\tilde{\boldsymbol{\theta}}_{n,b}) = 1/\text{rank}_{\tilde{\boldsymbol{\theta}}_{n,b}}(\boldsymbol{s}_{I_{n,b}})$
      if $\tilde{r} > r_{n,b}^{\text{mod}}$ **then**
        $\boldsymbol{\theta}_{n,b} \leftarrow \boldsymbol{\theta}_{n,b} - \gamma(\tilde{r} - r_{n,b}^{\text{mod}})\boldsymbol{q}$
      end if
    end for
    $\boldsymbol{\theta}_{n,b+1} \leftarrow \boldsymbol{\theta}_{n,b}$
  end for
  $\boldsymbol{\theta}_{n+1,1} \leftarrow \boldsymbol{\theta}_{n,B}$
  // Offline Training
  $\mathcal{D}_{n+1,0} \leftarrow \mathcal{D}_{n,B}$
  Update user attention model $\boldsymbol{\beta}_{n+1} \leftarrow \Delta(\boldsymbol{\beta}_n)$ on $\mathcal{D}_{n+1,0}$
end for

6 EXPERIMENTS

In this section, we empirically study the proposed NDB by addressing the following research questions:

RQ1: How does the proposed NDB perform in comparison with state-of-the-art baselines?

RQ2: Whether NDB is efficient enough to ensure the real-time requirement of online recommendations?

RQ3: How is NDB impacted by the number of dueling models K?

RQ4: What is the impact of major components of NDB on the recommendation performance?

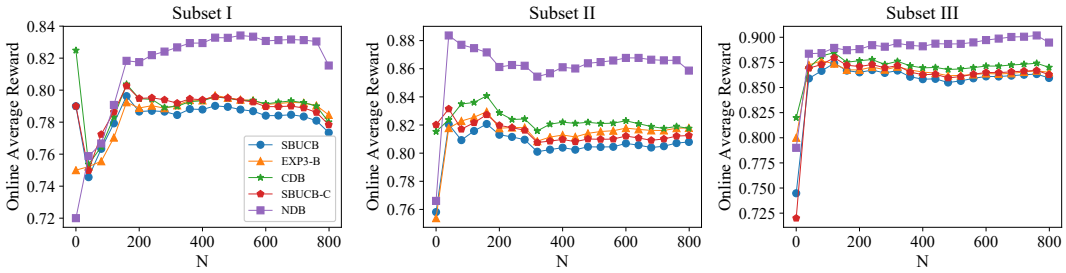

Figure 5: Online average rewards of SBUCB, EXP3-B, CDB, SBUCB-C, and the proposed NDB on first three subsets from `Last.fm 1K` dataset. Results of the last three subsets can be found in Appendix B.4.

6.1 Experimental Settings

6.1.1 Dataset. For evaluation, we used the `Last.fm 1K` dataset [3] (with multiple subsets), containing approximately 20 million listening events of 992 distinct Last.fm users between July 2005 and May 2009[1]. We enriched the original dataset using the Last.fm API[2], Spotify API[3], and MusicBrainz API[4], by which the missing information (e.g., duration) of candidate songs was completed. Specifically, `Last.fm 1K` consists of six subsets that each covers six months' worth of user-song interactions [21, 26], and the interactions collected in 2009 were used for testing the online simulator.

For the observed user feedbacks, following the standard practice in [21, 22], we used the *playrate* to distinguish the skip feedbacks from listening events. Specifically, as in [22], we treated an event as a negative feedback (i.e., a skip) if the playrate of the song was less than 90% and others as positive feedbacks (i.e., a play).

Following the settings in [21, 40], we represented each user and song using low-dimensional features, and the context as a 120-dimensional vector $\boldsymbol{x}_n(\boldsymbol{s})$ defined in Section 3.1, i.e., $d_x = 120$.

More details about the dataset and data preparation can be found in Appendix B.1.

6.1.2 Baselines. NDB was compared with several algorithms that directly use the unmodified user feedbacks, including:

SBUCB [16] is a batched version of LinUCB [19], which updates the policy after receiving a batch of feedback data, where LinUCB is a classic linear contextual bandit approach for recommendation.

EXP3-B [20]: EXP3 is a classic algorithm for adversarial bandits [5], where the agent chooses an action according to a distribution constructed by the exponential weights. The batched and contextual version of EXP3 is denoted by EXP3-B.

CDB [21] is a state-of-the-art bandit optimization algorithm for music streaming recommendation that updates the model online via multiple duels. Moreover, CDB is a personalized model that learns different recommendation models for different users.

NDB was also compared to **SBUCB with Cutoff (SBUCB-C)** that addresses the attention bias by heuristically discarding the user positive feedbacks that may be false. It is a variant of SBUCB that equips with a play cutoff technology [25], where only the first two songs played entirely after the user's skip action are treated as the positive user feedbacks, and the rest of user feedbacks are ignored.

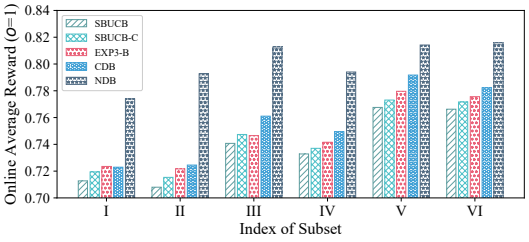

Figure 6: Online average reward for the case that the user attention $o = 1$ on six subsets from the `Last.fm 1K` dataset.

6.1.3 Evaluation Protocol. In streaming recommendation, we cannot guarantee that each recommended song to the user had a corresponding feedback in the log data. To address this issue, following [18, 25, 40], all the algorithms were tested in simulated online environments trained on log data accurately. More details regarding the online environments can be found in Appendix B.2. Briefly, at each step, the online environment received a song from the recommendation model, and generated the attention and relevance score to return user feedback (1 = "play", 0 = "skip") according to Table 2.

Following these settings, we conducted online tests on Subset I–Subset VI to assess the effectiveness and efficiency of the music streaming recommenders. To this end, we define the *online average reward* up to the first n episodes as $\frac{1}{nB}\sum_{k=1}^{n}\sum_{b=1}^{B} r_{k,b}$, where $r_{k,b}$ is the observed user feedback at step b in the k-th episode. In Table 3, we also reported the online average reward w.r.t. all user sessions.

6.2 Experiment Results and Analyses

This section addresses the four research questions stated earlier.

6.2.1 ***RQ1: Comparison against Baselines.*** Figure 5 reports the average reward of the baselines and the proposed NDB on the `Last.fm 1K` dataset. We can observe that NDB achieves the highest average reward after running about 200 sessions on all subsets. Table 3 shows the average reward w.r.t. all user sessions and running time for NDB and the baselines on six subsets. The proposed algorithm outperformed all the baselines on all six subsets in terms of the rewards, verifying the effectiveness of NDB for counteracting user attention bias in streaming recommendation.

We also compared all the algorithms in terms of the online average rewards when users paid attention to the recommended songs (i.e., attention variable $o = 1$). The results in Figure 6 showed that

[1]Note that there exist several other readily available datasets for music streaming recommendation [3, 9, 32]. However, they cannot support the investigations in this paper because these datasets either did not record listening history in timestamp units or did not consist of user side information.
[2]http://www.last.fm
[3]https://developer.spotify.com/documentation/web-api/
[4]http://www.musicbrainz.org/

Table 3: Comparisons of online average reward (w.r.t. all user sessions) and running time for NDB and baselines on six subsets from `Last.fm 1K`. '$*$': improvements over baselines are statistical significant (t-test, p-value< 0.05). Recommendation: online recommendation. The running time of the online process means the time cost of recommendation or training at each step for one song, while the running time of offline training denotes the time cost of model training at the end of each user session (i.e., each episode). In particular, Since CDB is a fully-online algorithm, it does not have any time cost of offline training.

Algorithm	Online average reward						Running time (sec., mean ± std)		
	Subset I	Subset II	Subset III	Subset IV	Subset V	Subset VI	Recommendation	Online training	Offline training
SBUCB [16]	0.7809	0.8062	0.8638	0.8729	0.8893	0.8921	1.96e-4±6.64e-5	2.65e-6±4.79e-7	6.88e-4±1.28e-4
EXP3-B [20]	0.7889	0.8127	0.8664	0.8754	0.8931	0.8953	2.92e-4±8.63e-5	2.55e-6±4.75e-7	8.09e-4±2.48e-4
CDB [21]	0.7892	0.8186	0.8704	0.8792	0.8982	0.9001	2.15e-5±1.94e-5	4.91e-4±6.91e-5	-
SBUCB-C [1, 16]	0.7859	0.8111	0.8673	0.8749	0.8919	0.8945	1.98e-4±7.47e-5	2.69e-6±6.09e-7	6.21e-4±1.42e-4
NDB (Ours)	**0.8248***	**0.8500***	**0.8963***	**0.9002***	**0.9113***	**0.9153***	3.67e-4±7.85e-5	4.38e-3±1.57e-3	1.35e-2±9.12e-4

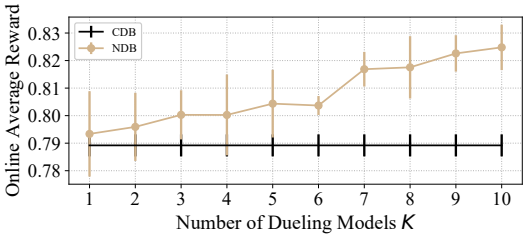

Figure 7: Impact of the number of dueling models K in NDB on Subset I, where the shaded area represents the standard deviation.

NDB also achieved the highest rewards when $o = 1$, indicating that reward modification can help enhance the prediction accuracy of user preferences.

6.2.2 ***RQ2: Running Time.*** In streaming recommendation scenarios, running time is another important metric. We reported the running time of online and offline processes in Table 3. Note that we treated a user feedback as a negative one if the playrate of the song was less than 90%, which means that we had 10% of a song's duration to complete online model training and recommendation of the next song. From the results of NDB, we observed that the running time of both the online recommendation and the online training at each step was on the order of milliseconds (ms), indicating that NDB met the real-time requirements in streaming recommendations. Besides, the running time of offline training for updating the attention model after each session was also acceptable.

6.2.3 ***RQ3: Impact of Number of Dueling Models*** K. The number of dueling models K in NDB balanced the effectiveness and efficiency of the online training process. Specifically, a larger K will explore more updating directions of the current relevance prediction model and consequently enhance the effectiveness. On the other hand, as the number of dueling models increased, the time consumption of NDB also increased according to the complexity analysis described in Section 4.5.3. To verify the analyses, we conducted experiments by varying K from 1 to 10 on Subset I. Figure 7 illustrated the performances of different K in terms of the online average rewards w.r.t. all user sessions. The results, as expected, indicated a beneficial influence on effectiveness while raising the value of K. Compared with NDB using one dueling model per user feedback (i.e., $K = 1$), NDB with $K = 10$ can get approximately 3% improvement on performance. Besides, we achieved similar observations on other subsets in experiments. These results further verified the effectiveness of the online training process with the modified rewards in NDB.

Table 4: Ablation study on Subset I.

Algorithm	Online average reward
NDB w/o RWNN	0.8049
NDB w/o modification	0.8102
NDB	**0.8248**

6.2.4 ***RQ4: Ablation Study.*** NDB consists of two important components: one is RWNN encoding candidate songs for better prediction of the relevance probability, the other is the reward modification component estimating the expected true reward for better online training. To address the fourth research question, we performed ablation studies by omitting some of the major components of NDB on Subset I. These NDB variations include: (a) predicting the relevance probability without using RWNN (denoted by 'NDB w/o RWNN'), and (b) online training without reward modification (denoted by 'NDB w/o modification'). From the results reported in Table 4, we observed that compared with the original NDB, the performances of both NDB variations dropped, demonstrating the effectiveness of both components. More importantly, ablation studies on all other subsets had similar conclusions. These ablation studies clearly showed that both RWNN and the reward modification were crucial parts of the NDB to achieve its potential.

7 CONCLUSION

This paper aims to eliminate the user attention bias and enhance the recommendation performance in music streaming recommendation. Specifically, we propose an online learning approach called Neural Dueling Bandit (NDB). The proposed NDB captures the mechanisms of user attention and users' music preference using neural networks, achieves an unbiased estimate of true rewards, and enjoys a sub-linear regret upper bound against the optimal policy for online recommendation. Experimental results demonstrated the effectiveness and efficiency of NDB in steaming recommendation.

ACKNOWLEDGMENTS

This work was funded by the National Key R&D Program of China (2019YFE0198200), the National Natural Science Foundation of China (62006234, 61872338, 61832017), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China (No.22XNH027), and Public Policy and Decision-making Research Lab of Renmin University of China. We would like to thank Bruno L. Pereira and Ahmed Kachkach for their valuable suggestion and help of this work.

REFERENCES

[1] Kachkach Ahmed. 2016. *Analyzing user behavior and sentiment in music streaming services.* Master's thesis. KTH Royal Institute of Technology.

[2] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 385–394.

[3] áOscar Celma. 2010. *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space.* Springer Publishing Company, Incorporated.

[4] Walid Bendada, Guillaume Salha, and Théo Bontempelli. 2020. Carousel personalization in music streaming apps with contextual bandits. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 420–425.

[5] Ilai Bistritz, Zhengyuan Zhou, Xi Chen, Nicholas Bambos, and Jose Blanchet. 2019. Online EXP3 learning in adversarial bandits with delayed feedback. In *Advances in Neural Information Processing Systems 32*. 11349–11358.

[6] Geoffray Bonnin and Dietmar Jannach. 2014. Automated generation of music playlists: Survey and experiments. *Comput. Surveys* 47, 2 (2014), 1–35.

[7] Léon Bottou, Jonas Peters, Joaquin Quiñonero Candela, Denis Xavier Charles, Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Y. Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research* 14 (2013), 3207–3260.

[8] Léa Briand, Guillaume Salha-Galvan, Walid Bendada, Mathieu Morlon, and Viet-Anh Tran. 2021. A semi-personalized system for user cold start recommendation on music streaming apps. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2601–2609.

[9] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The music streaming sessions dataset. In *Proceedings of the 2019 World Wide Web Conference*. 2594–2600.

[10] Zhiyong Cheng, Shen Jialie, and Steven C.H. Hoi. 2016. On effective personalized music retrieval by exploring online user behaviors. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 125–134.

[11] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1724–1734.

[12] Bruce Ferwerda and Markus Schedl. 2016. Personality-based user modeling for music recommender systems. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 254–257.

[13] Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. 2005. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*. 385–394.

[14] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[15] Kartik Gupta, Noveen Sachdeva, and Vikram Pudi. 2018. Explicit modelling of the implicit short term user preferences for music recommendation. In *Proceedings of the 40th European Conference on Information Retrieval Research*. 333–344.

[16] Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose H. Blanchet, Peter W. Glynn, and Yinyu Ye. 2020. Sequential batch learning in finite-action linear contextual bandits. *CoRR* abs/2004.06321 (2020).

[17] Elad Hazan. 2016. Introduction to online convex optimization. *Foundations and Trends in Optimization* 2, 3-4 (2016), 157–325.

[18] Olivier Jeunen, David Rohde, Flavian Vasile, and Martin Bompaire. 2020. Joint policy-value learning for recommendation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1223–1233.

[19] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 661–670.

[20] Gergely Neu and Julia Olkhovskaya. 2020. Efficient and robust algorithms for adversarial linear contextual bandits. In *Proceedings of the 33rd Conference on Learning Theory*. 3049–3068.

[21] Bruno L Pereira, Alberto Ueda, Gustavo Penha, Rodrygo LT Santos, and Nivio Ziviani. 2019. Online learning to rank for sequential music recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 237–245.

[22] Joachim Valdemar Yde Peter Vergerakis. 2020. *Exploring skips and long-Term preferences in session-based music recommendation.* Master's thesis. Aalborg University.

[23] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*. 1177–1184.

[24] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. 2020. What's hidden in a randomly weighted neural network?. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11890–11899.

[25] Reza Reza Aditya Permadi. 2018. *Improving recommender systems algorithms for personalized music video television by incorporating user consumption behaviour and multiple types of user feedback.* Master's thesis. Delft University of Technology.

[26] Noveen Sachdeva, Kartik Gupta, and Vikram Pudi. 2018. Attentive neural architecture incorporating song features for music recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 417–421.

[27] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. 457–466.

[28] Shai Shalev-Shwartz. 2011. Online learning and online convex optimization. *Foundations and Trends in Machine Learning* 4, 2 (2011), 107–194.

[29] Yading Song, Simon Dixon, and Marcus Pearce. 2012. A survey of music recommendation systems and future perspectives. In *Proceedings of the 9th International Symposium on Computer Music Modeling and Retrieval*. 395–410.

[30] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. 2007. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research* 8 (2007), 985–1005.

[31] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems 33* (2020), 7537–7547.

[32] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 2015. 30Music listening and playlists dataset. In *RecSys Posters*. 75.

[33] Andreu Vall. 2015. Listener-inspired automated music playlist generation. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 387–390.

[34] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*. 2643–2651.

[35] Sergey Volokhin and Eugene Agichtein. 2018. Understanding music listening intents during daily activities with implications for contextual music recommendation. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*. 313–316.

[36] Nils Wlömert and Dominik Papies. 2016. On-demand streaming services and music industry revenues—Insights from Spotify's market entry. *International Journal of Research in Marketing* 33, 2 (2016), 314–327.

[37] Shota Yasui, Gota Morishita, Komei Fujita, and Masashi Shibata. 2020. A feedback shift correction in predicting conversion rates under delayed feedback. In *Proceedings of the Web Conference 2020*. 2740–2746.

[38] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1201–1208.

[39] Boxun Zhang, Gunnar Kreitz, Marcus Isaksson, Javier Ubillos, Guido Urdaneta, Johan A Pouwelse, and Dick Epema. 2013. Understanding user behavior in spotify. In *Proceedings of the IEEE INFOCOM 2013*. 220–224.

[40] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. 2021. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 41–50.

[41] Xiao Zhang and Shizhong Liao. 2019. Incremental randomized sketching for online kernel learning. In *Proceedings of the 36th International Conference on Machine Learning*. 7394–7403.

A APPENDIX: DETAILED PROOFS

A.1 Proof of Theorem 4.1

Proof of Theorem 4.1. From the definitions of the modified reward $r^{\text{mod}}(\boldsymbol{x}_n(\boldsymbol{s}), c)$ in Eq.(3) and the importance weight in Eq.(4), we have

$$\begin{aligned}\mathbb{E}_c[r^{\text{mod}}(\boldsymbol{x}_n(\boldsymbol{s}), c)] &= w \cdot \mathbb{E}_c[r(\boldsymbol{x}_n(\boldsymbol{s}), c)] \\ &= w \cdot \Pr\{c = 1 |\ \boldsymbol{x}_n(\boldsymbol{s})\} \\ &= \frac{\Pr\{v = 1 |\ \boldsymbol{x}_n(\boldsymbol{s})\}}{\Pr\{c = 1 |\ \boldsymbol{x}_n(\boldsymbol{s})\}} \cdot \Pr\{c = 1 |\ \boldsymbol{x}_n(\boldsymbol{s})\} \\ &= \mathbb{E}_v\left[r(\boldsymbol{x}_n(\boldsymbol{s}), v)\right].\end{aligned}$$

□

A.2 Proof of Theorem 5.1

Proof of Theorem 5.1. From lemma 2.1 in [13], we obtain that

$$\nabla \hat{\ell}_{n,b}(\boldsymbol{\theta}) = \begin{cases} \mathbb{E}_{\boldsymbol{q}\in\mathbb{U}_s}\left\{\frac{d}{\delta}\left[\tilde{r}(\boldsymbol{\theta}+\delta\boldsymbol{q}) - r_{n,b}^{\text{mod}}\right]\boldsymbol{q}\right\} & \tilde{r}(\boldsymbol{\theta}) > r_{n,b}^{\text{mod}}, \\ 0 & \tilde{r}(\boldsymbol{\theta}) \le r_{n,b}^{\text{mod}}. \end{cases} \tag{7}$$

From the regret analysis of online linear optimization using online gradient descent [17, 28], we have

$$\begin{aligned}&\sum_{n\in[N],b\in[B]} \left\langle [\tilde{r}(\boldsymbol{\theta}_{n,b}+\delta\boldsymbol{q}) - r_{n,b}^{\text{mod}}]\boldsymbol{q}, \boldsymbol{\theta}_{n,b} - \boldsymbol{\theta}^* \right\rangle \\ &\le \frac{\|\boldsymbol{\theta}^*\|_2^2}{2\gamma} + \gamma \sum_{n\in[N],b\in[B]} \left\| \left[\tilde{r}(\boldsymbol{\theta}_{n,b}+\delta\boldsymbol{q}) - r_{n,b}^{\text{mod}}\right]\boldsymbol{q} \right\|_2^2.\end{aligned} \tag{8}$$

Multiplying both sides of Eq.(8) by d/δ and taking expectation, from Eq.(7) we get

$$\begin{aligned}&\mathbb{E}\left[\sum_{n\in[N],b\in[B]} \left\langle \nabla\hat{\ell}(\boldsymbol{\theta}_{n,b}), \boldsymbol{\theta}_{n,b} - \boldsymbol{\theta}^* \right\rangle\right] \\ &\le \frac{d}{2\gamma\delta}\|\boldsymbol{\theta}^*\|_2^2 + \frac{d\gamma}{\delta} \sum_{n\in[N],b\in[B]} \mathbb{E}\left\| \left[\tilde{r}(\boldsymbol{\theta}_{n,b}+\delta\boldsymbol{q}) - r_{n,b}^{\text{mod}}\right]\boldsymbol{q} \right\|_2^2 \\ &\le \frac{d}{2\gamma\delta}C_{\boldsymbol{\theta}} + \frac{d\gamma}{\delta} \sum_{n\in[N],b\in[B]} \mathbb{E}\left[\tilde{r}(\boldsymbol{\theta}_{n,b}+\delta\boldsymbol{q}) - r_{n,b}^{\text{mod}}\right]^2 \\ &\le \frac{d}{2\gamma\delta}C_{\boldsymbol{\theta}} + \frac{d\gamma}{\delta}T.\end{aligned} \tag{9}$$

From the condition of Lipschitz continuous gradients of $\hat{\ell}$, we obtain

$$\begin{aligned}\hat{\ell}_{n,b}(\boldsymbol{\theta}_{n,b}) - \hat{\ell}_{n,b}(\boldsymbol{\theta}^*) &\le \langle \nabla\hat{\ell}_{n,b}(\boldsymbol{\theta}_{n,b}), \boldsymbol{\theta}_{n,b} - \boldsymbol{\theta}^* \rangle + \frac{L_{\text{g}}}{2}\|\boldsymbol{\theta}_{n,b} - \boldsymbol{\theta}^*\|_2^2 \\ &\le \langle \nabla\hat{\ell}_{n,b}(\boldsymbol{\theta}_{n,b}), \boldsymbol{\theta}_{n,b} - \boldsymbol{\theta}^* \rangle + 2\delta C_{\boldsymbol{\theta}}.\end{aligned} \tag{10}$$

Combining Eq.(10) with Eq.(9) yields

$$\mathbb{E}\left[\sum_{n\in[N],b\in[B]} \left(\hat{\ell}_{n,b}(\boldsymbol{\theta}_{n,b}) - \hat{\ell}_{n,b}(\boldsymbol{\theta}^*)\right)\right] \le \frac{d}{2\gamma\delta}C_{\boldsymbol{\theta}} + \frac{d\gamma}{\delta}T + 2\delta C_{\boldsymbol{\theta}}T. \tag{11}$$

Since $\tilde{r}(\boldsymbol{\theta})$ is Lipschitz continuous with Lipschitz constant L, we have $|\tilde{r}(\boldsymbol{\theta}+\delta\boldsymbol{u}) - \tilde{r}(\boldsymbol{\theta})| \le L\delta\|\boldsymbol{u}\|_2$. Furthermore, when $\tilde{r}(\boldsymbol{\theta}) > r_{n,b}^{\text{mod}}$, since $\|\boldsymbol{u}\|_2 \le 1$, we obtain

$$\begin{aligned}|\ell_{n,b}(\boldsymbol{\theta}+\delta\boldsymbol{u}) - \ell_{n,b}(\boldsymbol{\theta})| &= \left|[\tilde{r}(\boldsymbol{\theta}+\delta\boldsymbol{u}) - r_{n,b}^{\text{mod}}] - [\tilde{r}(\boldsymbol{\theta}) - r_{n,b}^{\text{mod}}]\right| \\ &= |\tilde{r}(\boldsymbol{\theta}+\delta\boldsymbol{u}) - \tilde{r}(\boldsymbol{\theta})| \\ &\le L\delta\|\boldsymbol{u}\|_2 \\ &\le L\delta,\end{aligned}$$

yielding that, when $\tilde{r}(\boldsymbol{\theta}) > r_{n,b}^{\text{mod}}$,

$$|\hat{\ell}_{n,b}(\boldsymbol{\theta}) - \ell_{n,b}(\boldsymbol{\theta})| \le L\delta. \tag{12}$$

Combining Eq.(11) with Eq.(12) yields

$$\begin{aligned}&\mathbb{E}\left[\text{Reg}(\{\boldsymbol{\theta}_{n,b}\}_{n\in[N],b\in[B]}, \boldsymbol{\theta}^*)\right] \\ &\le \mathbb{E}\left[\sum_{n\in[N],b\in[B]} \left(\hat{\ell}_{n,b}(\boldsymbol{\theta}_{n,b}) - \hat{\ell}_{n,b}(\boldsymbol{\theta}^*)\right)\right] + 2L\delta T \\ &\le \frac{d}{2\gamma\delta}C_{\boldsymbol{\theta}} + \frac{d\gamma}{\delta}T + 2(C_{\boldsymbol{\theta}} + L)\delta T.\end{aligned}$$

□

B APPENDIX: EXPERIMENT DETAILS

B.1 Data Preparation

As shown in Table 5, `Last.fm 1K` consists of six subsets that each covers six months' worth of user-song interactions (i.e., events) [21, 26], and the interactions collected in 2009 were used for testing the online simulator. After enrichment and filtering, the pre-processed `Last.fm 1K` contained 984 unique users, 584, 194 unique songs and 10,840,109 interactions. User sessions were split as follows: two consecutive listening events were considered to be in the same user session if they occurred within 60 minutes of each other.

Each instance in the original `Last.fm 1K` dataset only contained the userID, event timestamp, artistID, songID, and song title, and specific user feedback (play or skip) was not provided. Following existing empirical studies [21, 22], we used *playrate* to distinguish the skip feedback which refers to the proportion of how much the recommended song had been played (i.e, playtime of a song), compared to the song's total duration. To obtain song durations for computing playrates, we captured the missing information of data from the Last.fm API[5], Spotify API[6] and MusicBrainz API[7]. Specifically, we computed the playrate of a song by comparing the song's duration with the number of seconds between the starting points of the current and the next song. Following standard practice [22], we treated a user feedback as a negative one (i.e., a skip) if the playrate of the song is less than 90% or a positive one (i.e., a play) otherwise.

Following the settings in [21, 40], we represented each user and song using low-dimensional features for efficient online recommendations. Specifically, we used the feature hashing to mapping ID type features (such as userID and songID). Besides, the categorical features were represented as one-hot vectors and then concatenated to the dense features obtained from the Spotify API (e.g., acousticness, speechiness, danceability, energy, etc.). Then, by conducting

[5] http://www.last.fm
[6] https://developer.spotify.com/documentation/web-api/
[7] http://www.musicbrainz.org/

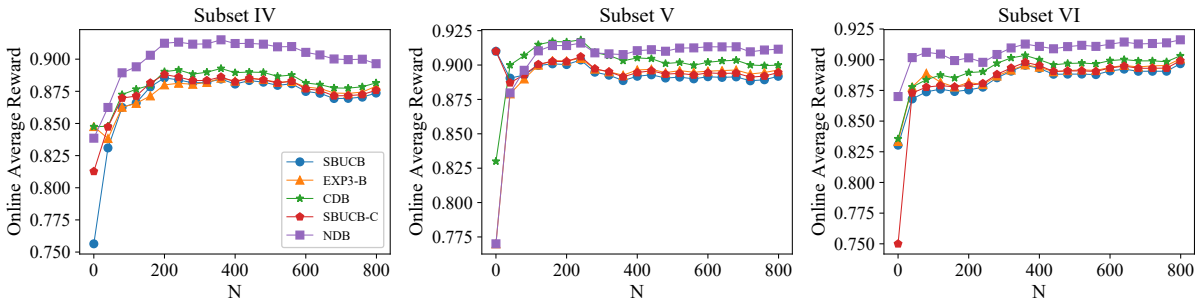

Figure 8: Online average rewards of SBUCB, EXP3-B, CDB, SBUCB-C, and the proposed NDB on last three subsets from `Last.fm 1K`.

Table 5: Statistics of the pre-processed `Last.fm 1K dataset`.

Subset	Date	#Users	#Songs	#Sessions	#Events	Skip (%)
I	2006/01-2006/06	418	168,010	63,574	1,288,457	22.89
II	2006/07-2006/12	538	198,353	88,040	1,707,884	18.83
III	2007/01-2007/06	622	222,788	97,863	1,858,454	15.70
IV	2007/07-2007/12	654	250,926	102,988	1,849,446	12.48
V	2008/01-2008/06	706	269,237	108,429	1,977,493	12.12
VI	2008/07-2008/12	759	288,715	117,235	2,158,375	13.11

Table 6: Ranges of the hyper-parameter tuning in the proposed NDB.

Component	Hyper-parameter	Range
Attention prediction	Learning rate in Adam optimizer	{1e-1, 1e-2, 1e-3, 1e-4}
	Weight decay in Adam optimizer	{1e-2, 1e-3, 1e-4, 1e-5}
	Clipping threshold $w_{\min}$	$[0.1 : +0.1 : 0.5]$
Relevance prediction	Dimension d of $\boldsymbol{\theta}$	$\{2^i, i = [6 : +1 : 9]\}$
	Inverse of the standard deviation $1/\xi$	$\{2^i, i = [-8 : +1 : 8]\}$
	Stepsize γ	$[0.001 : 100]$
	Exploration parameter δ	$[0.001 : 100]$
	Number of dueling models K	$[1 : +1 : 10]$

principal component analysis to these feature vectors and concatenating to the hashed features, we obtained a 30-dimensional vector representation for each user $\boldsymbol{u}_n$ ($d_u = 30$) and a 90-dimensional vector representation for each song $\boldsymbol{s}$ ($d_s = 90$), yielding a 120-dimensional feature vector $\boldsymbol{x}_n(\boldsymbol{s})$ at each step($d_x = 120$).

B.2 Simulated Online Environment

In this study, we focused on a streaming playlist generation scenario, where songs were dynamically recommended once at a time by taking into account the historical user interactions during their listening session. However, in practice, user feedbacks were only available for the songs actually consumed by the users when conducting online recommendation. In such an online scenario, we cannot guarantee that each song recommended by the model to the user had a corresponding feedback in the log data. Instead, to ensure our evaluation reliable, following [18, 25, 40], we used a simulated online environment. More specifically, the simulated online environment consists of two models: a model for simulating the user attention and a model for outputting the true user preference (i.e., the relevance). The user attention simulator was an exponential decay model in [1], where the user's attention probability decayed with time until another active user action happens (i.e., skip action) or failed to a nil value. We trained the relevance model by DeepFM [14], which is a popular CTR prediction model that can return a relevance score given a user and a song. DeepFM was trained on six filtered subsets and tested on the log data collected in 2009, and its AUCs in testing were over 77%, assuring that the online environment can provide nearly realistic feedbacks of users. At each step, the online environment receives a song from the recommend model, and generate the attention and relevance score to return user feedback (1 = "play", 0 = "skip") according to the relations in Table 2.

To evaluate the performances of online recommendation, we only retained the user sessions with more than 100 user-song interactions in the log data, where the songs were actually played by the user with realistic feedback (positive or negative). To maintain the streaming nature of data in streaming recommendation, the environment kept the order of timestamps for each interaction. In the candidate song set, following practice in [21], we chose 99 extra songs at random from a set of 1,000 songs that were most similar to the first song in this session. In this way, we reduced the original problem into a problem of online learning to rank: at each step, the recommendation model must pick one song from the 100 candidate songs for generating a list of songs in an online streaming manner.

B.3 More Implementation Details

All the experiments were run on a Linux workstation with the Intel(R) Xeon(R) CPU E5-2630 v4 (2.20GHz) and a single NVIDIA Tesla P100 GPU. To enhance the effect of reward modification, we applied a clipping technique to importance weights in Eq.(4) as $w = \min\{w, w_{\min}\}$, where $w_{\min} \in (0, 1]$ denotes the clipping threshold. For the stepsize γ and the exploration parameter δ, we multiplied δ and γ by $\|\boldsymbol{\theta}_{1,1}\|_2$, since the theoretical optimal values of δ and γ were proportional to the upper bound of $\|\boldsymbol{\theta}\|_2, \boldsymbol{\theta} \in \Omega$, shown in Remark 1. The ranges of all the hyper-parameter tuning were listed in Table 6.

B.4 Results of Last Three Subsets

As shown in Figure 8, the proposed NDB still outperformed the baselines on the last three subsets.