

CUPID: A Real-Time Session-Based Reciprocal Recommendation System for a One-on-One Social Discovery Platform

Beomsu Kim^{*1}, Sangbum Kim^{*1}, Minchan Kim^{*1}, Joonyoung Yi¹, Sungjoo Ha¹
Suhyun Lee¹, Youngsoo Lee¹, Gihoon Yeom¹, Buru Chang², Gihun Lee^{†1}
¹Hyperconnect, ²Sogang University

Abstract—This study introduces CUPID, a novel approach to session-based reciprocal recommendation systems designed for a real-time one-on-one social discovery platform. In such platforms, low latency is critical to enhance user experiences. However, conventional session-based approaches struggle with high latency due to the demands of modeling sequential user behavior for each recommendation process. Additionally, given the reciprocal nature of the platform, where users act as items for each other, training recommendation models on large-scale datasets is computationally prohibitive using conventional methods. To address these challenges, CUPID decouples the time-intensive user session modeling from the real-time user matching process to reduce inference time. Furthermore, CUPID employs a two-phase training strategy that separates the training of embedding and prediction layers, significantly reducing the computational burden by decreasing the number of sequential model inferences by several hundredfold. Extensive experiments on large-scale Azar datasets demonstrate CUPID’s effectiveness in a real-world production environment. Notably, CUPID reduces response latency by more than 76% compared to non-asynchronous systems, while significantly improving user engagement.

Index Terms—Session-based Recommendation, Reciprocal Recommendation, Real-time One-on-one Social Discovery

I. INTRODUCTION

Azar is a leading real-time social discovery platform that connects users for one-on-one video conversations. To facilitate these interactions, the platform gathers users who signal their readiness for immediate video calls into a matching pool. The platform then matches users from this pool aiming to maximize overall user satisfaction, measured by the total chat duration across all pairs. Longer chat durations are indicative of more engaging and satisfying interactions, thus serving as a proxy for user satisfaction. In such reciprocal recommendation systems, where both users need to be mutually satisfied, the recommendations must reflect the preferences of both parties [1–3]. Furthermore, as users engage with the platform, their preferences can change dynamically [4, 5]. For example, a user might start by wanting to chat casually about favorite hobbies but later seek deeper conversations about social issues.

In real-time social discovery platforms, adapting to evolving user preferences is crucial for maintaining engagement and

satisfaction. One effective approach is session-based recommendations [6–8], where a session represents a single visit or interaction period during which the user actively engages with the platform. By focusing solely on the current session, session-based recommendations consider a user’s behavior within that session rather than building a user profile from long-term historical data. This approach leverages session-specific information, enabling the system to respond to dynamic preferences [9–13] and address the cold-start problem [14–18], where new users lack sufficient historical data, by relying on data from the current session.

However, applying session-based recommendations to reciprocal recommendation systems with strict real-time constraints presents unique and significant challenges. First, conventional session-based systems build user profiles through computationally intensive session modeling [6, 19–21], which can take several seconds and thereby far exceeding the immediate response times required by platforms like Azar. This delay results in a bottleneck in delivering timely recommendations. Second, user behavior in reciprocal systems can evolve rapidly within a single session, even after each interaction. For example, a positive interaction might make a user more inclined toward similar profiles, while a negative experience could shift their preferences entirely. Moreover, conventional session based recommendations mostly assume static item representations [6, 22, 23]. In reciprocal systems [2, 24, 25], however, both user preferences and the items (i.e., other users) change dynamically since users act as both consumers and items. Consequently, each interaction not only updates a user’s preferences but also impacts other users’ representations, complicating the recommendation algorithm. These factors make real-time session-based reciprocal recommendations more complex than conventional systems. The differences between conventional session-based recommendation and its application in real-time reciprocal recommendation are illustrated in Figure 1.

To address these challenges, we propose CUPID, a session-based reciprocal recommendation system specifically designed for real-time social discovery platforms. For the inference efficiency, CUPID aims to minimize the overall time consumption of the recommendation pipeline by decoupling it from the computationally intensive session modeling for each user. More specifically, CUPID adopts an *asynchronous* session

^{*} These authors contributed equally to this work.

[†] Corresponding to: Gihun Lee (dylan.l@hpcnt.com).

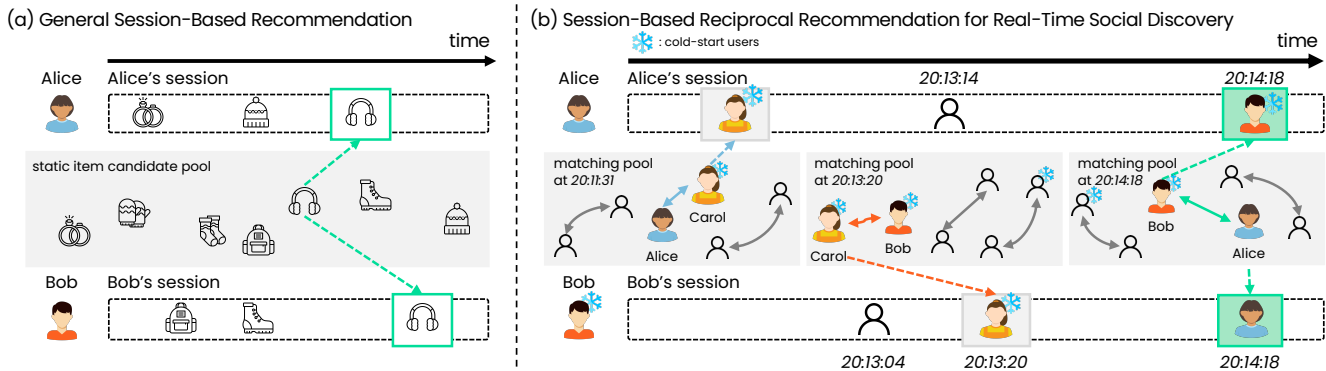


Fig. 1: Difference between (a) **conventional session-based recommendations** and (b) **session-based reciprocal recommendations** for real-time social discovery. The representation of the *earmuffs* remains unchanged for both *Alice* and *Bob*. In contrast, on real-time platforms, user representations continuously evolve with each session. For instance, after *Carol* interacts with both *Alice* and *Bob*, her representation changes based on the timing of these interactions. When *Bob* pairs with *Carol*, her representation now reflects her previous interaction with *Alice*.

modeling approach, where user session representations are updated separately from the recommendation process. In this approach, the asynchronously updated user profiles for session modeling are stored in a separate embedding memory. On the other hand, the feature embedding, which is computationally lightweight as it relies on static user information (e.g., country, gender) or match-related statistics, is updated synchronously. When a match request arrives, the system retrieves the pre-computed session embedding from the embedding memory and combines it with the synchronously computed feature embedding to estimate the chat duration between users.

To tackle the training complexity inherent in reciprocal environments, CUPID divides the training process into two distinct phases. In phase 1, the focus is on training the embedding layers that model user sessions and features. In phase 2, these embedding layers are frozen, and the prediction layer is trained to estimate chat duration using the pre-trained embeddings. This two-phase strategy reduces significantly the overall computational cost, which would otherwise be much higher if both components were trained jointly. By separating the training process, the embedding layer handles each user individually rather than modeling interactions between users for every match. This approach not only lowers the training cost but also ensures high prediction performance.

In our experiments, we evaluate CUPID using large-scale, real-world data from *Azar*. Both offline and online production tests demonstrate that CUPID significantly reduces the recommendation latency and enhances overall user satisfaction, proving its effectiveness for real-time reciprocal recommendation systems. Notably, implementing CUPID increases the average chat duration by 6.8% for warm-start users and 5.9% for cold-start users. At the same time, it reduces response latency by 79.7% for the 90-th percentile of users and 75.9% for the 99-th percentile in the *Azar* service.

Our main contributions are summarized as follows:

- We systematically formulate session-based reciprocal rec-

ommendation systems for real-time social discovery platforms. To the best of our knowledge, this is the first study to tackle this specific challenge. (Section II)

- We introduce CUPID, a novel session-based recommendation system for real-time reciprocal recommendation. Using asynchronous session embedding and a two-phase training strategy, CUPID improves both inference time and training efficiency. (Section III)
- We validate the efficacy of CUPID using large-scale real-world data from *Azar*. CUPID significantly enhances recommendation performance in both offline and online evaluations while meeting strict latency constraints required in real-time social discovery. (Section IV)

II. PROBLEM FORMULATION

A real-time social discovery platform connects online users enabling immediate, one-on-one conversations. Let \mathcal{U} represent the set of all users on such a platform. At any given time t , the matching pool $\mathcal{U}^{(t)} = \{u_1, u_2, \dots, u_n\}$ consists of n users available for matching. As illustrated in Figure 1, the matching pool $\mathcal{U}^{(t)}$ is dynamic, constantly changing as users log in or out and conversations begin or end. Each user $u_i \in \mathcal{U}^{(t)}$ is characterized by a set of features X_i (e.g., gender, country code, and other match-related statistics) and session information $S_i = [m_{i,1}, m_{i,2}, \dots, m_{i,h}]$, which includes h matching histories. Each matching history $m_{i,k} \in S_i$ consists of the chat counterpart u_j and the chat duration y_{ij} as follows:

$$\text{Matching History: } m_{i,k} = (u_j, y_{ij}). \quad (1)$$

The goal of the session-based reciprocal recommendation system is to optimally pair suitable users from $\mathcal{U}^{(t)}$ by considering their features and matching histories to maximize overall user satisfaction. We define a recommendation model $f(\cdot)$, which estimates satisfaction scores s_{ij} for all possible pairs of users (u_i, u_j) :

$$\text{Satisfaction Score: } s_{ij} = f(u_i, u_j). \quad (2)$$

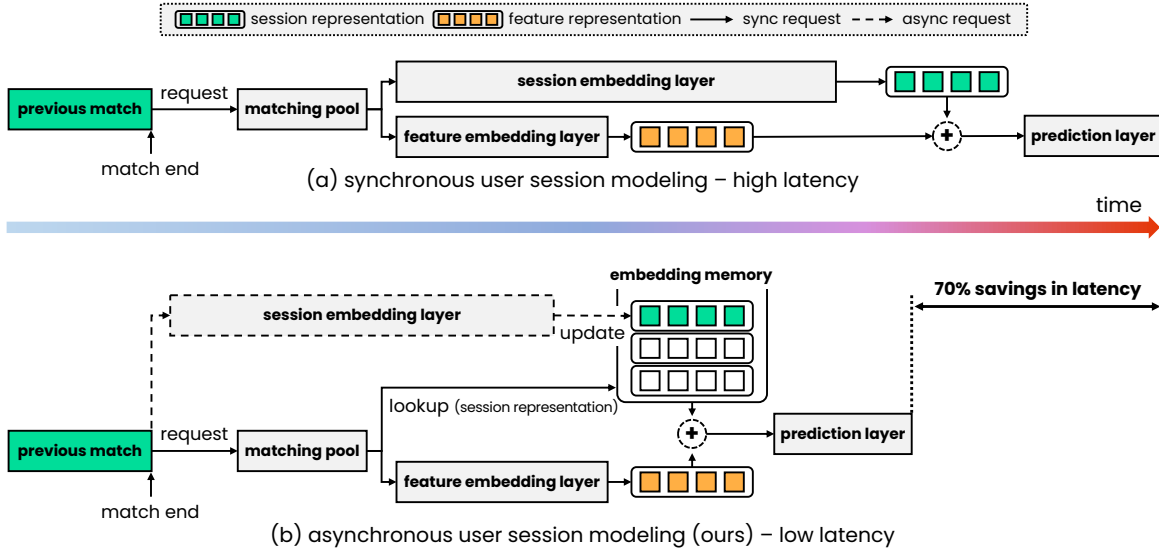


Fig. 2: System design consideration. (a) The overall latency of the session-based recommendation pipeline largely depends on the computational time of user session modeling (session embedding layer). (b) Our recommendation system, CUPID, reduces the latency of the recommendation pipeline by asynchronously conducting user session modeling in parallel with the pipeline.

For the satisfaction score, chat duration y_{ij} is used as a proxy for satisfaction scores, based on the assumption that longer conversations correlate with higher user satisfaction. Therefore, the recommendation model’s objective is revised to predict chat durations \hat{y}_{ij} for each user pair:

$$\text{Predicted Chat Duration: } \hat{y}_{ij} = f(u_i, u_j). \quad (3)$$

These predictions are used to connect users through efficient matching algorithms designed according to the service’s business logic. By predicting chat durations, the system can expedite connections that likely enhance user satisfaction, successfully addressing the challenges of dynamic user preferences in real-time social discovery platforms.

III. PROPOSED APPROACH: CUPID

In this section, we introduce CUPID, our session-based reciprocal recommendation system designed for real-world social discovery services with a focus on low-latency performance. We describe the implementation of Cupid and present a novel training method that efficiently captures mutual interests among users based on extensive matching histories.

A. System Design Considerations

As highlighted earlier, delivering recommendations with minimal delay is crucial for real-time social discovery platforms. Any latency may lead to longer wait times for users, harming user experience and potentially causing them to leave the service. A key challenge is efficiently modeling short-term, dynamic user behaviors to capture real-time preferences and intents. CUPID addresses two primary considerations: (i) rapidly computing satisfaction scores for all potential user pairs in the matching pool to minimize latency, and (ii) overcoming the slower processing times associated with sequence modeling architectures, such as RNNs or transformers. To

tackle these challenges, we have developed two core strategies for score computation and session modeling.

Linear Scaling Score Computation We compute the expected satisfaction score \hat{y}_{ij} (i.e., chat duration) by applying a simple linear transformation to the dot product of user representations as follows:

$$\hat{y}_{ij} = f(u_i, u_j) = w(\mathbf{e}_i \cdot \mathbf{e}_j) + b, \quad (4)$$

where \mathbf{e}_i and \mathbf{e}_j are the d -dimensional representation of users u_i and u_j , respectively. This approach allows us to compute the matrix of predicted satisfaction scores $\hat{\mathbf{Y}} \in \mathbb{R}^{n \times n}$ for all user pairs efficiently using a single matrix multiplication by leveraging optimized BLAS [26] libraries.

Asynchronous Session Modeling We decouple the computationally intensive user session modeling from the real-time matching pipeline by handling it asynchronously. This design significantly enhances the responsiveness of our recommendation system, enabling CUPID to deliver swift recommendations, which is essential for maintaining user engagement, as illustrated in Figure 2. An overview of Cupid’s architecture is provided in Figure 3. The performance of CUPID is measured by the *Mean Squared Error (MSE)* as follows:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|\mathcal{D}|} \sum_{m \in \mathcal{D}} (\hat{y}_{ij} - y_{ij})^2, \quad (5)$$

where \mathcal{D} is the dataset match history of all users. Further details of CUPID are presented in the subsequent sections.

B. Asynchronous Session Embedding Layer f_s

To ensure low-latency recommendations, CUPID models user behaviors in their sessions asynchronously rather than synchronously with matching requests. As illustrated in Figure 2(b), when a user u_i ’s previous match ends, the session

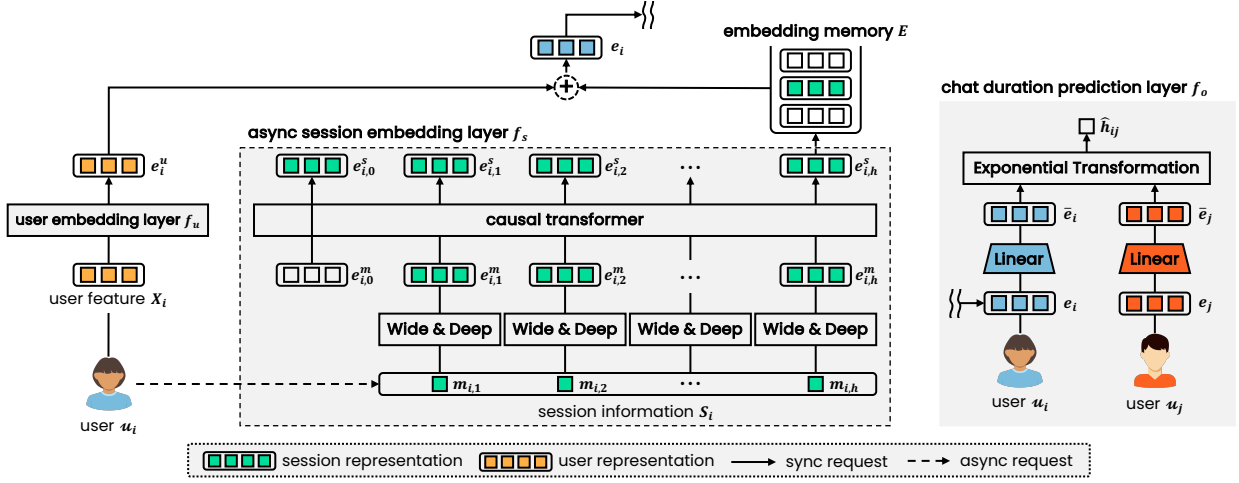


Fig. 3: An overview of CUPID architecture. The user u_i 's features X_i and session information S_i are modeled into the user feature representation e_i^u and the session representation e_i^s via the user feature embedding layer f_u and the session embedding layer f_s , respectively. The session representation is asynchronously computed and stored in the embedding memory E .

representation vector e_i^s is computed asynchronously using the session embedding layer f_s . More specifically, each matching history m in user u_i 's session information S_i is embedded into a representation e^m using Wide& Deep model [27]. This incorporates features X_i from the user u_j , and the features X_j from the chat counterpart user u_j , along with the chat duration y_{ij} . The user session representation e_i^s is then formed from these matching history representations $[e_{i,1}^m, e_{i,2}^m, \dots, e_{i,h}^m]$ employing a causal transformer, ensuring that each output $e_{i,k}^s$ represents the user's state after the k -th match, influenced only by preceding matches. The final session representation e_i^s is stored in an embedding memory E , replacing any existing representation. When user u_i requests a new match, the stored embedding e_i^s is retrieved to predict chat durations. Note that the session representation may not be updated before the session representation lookup occurs, as the computation might still be in progress when a new match is requested. In such cases, we refer to the session representation retrieved as a *delayed* session representation.

This design provides significant advantages: it decouples the slower user session modeling from the synchronous matching pipeline, improving both recommendation speed and efficiency. However, asynchronously updating session representations may cause recent information to be displaced during inference, as new match data could arrive while the session representations are still being updated. Despite this, the system incurs only a few seconds of delay, so the impact on performance is negligible. Furthermore, by handling session information asynchronously, the overall throughput of session processing is enhanced through the batching of multiple inferences, which also reduces computational costs.

C. Synchronous User Feature Embedding Layer f_u

Along with session information, Cupid incorporates user features such as demographic details (e.g., gender, country) and other match statistics to capture general user preferences.

We use Wide&Deep [27] as the user feature embedding layer f_u , which processes the user features X_i to generate a representation $e_i^u = f_u(X_i)$. This representation e_i^u is then used in the prediction layer to estimate chat duration of users.

D. Chat Duration Prediction Layer f_o

The chat duration prediction layer f_o aims to accurately predict the chat duration for a user pair (u_i, u_j) by combining their session and feature representations:

$$e_i = e_i^s + e_i^u, \quad e_j = e_j^s + e_j^u. \quad (6)$$

While a simple method to predict chat duration may involve computing the dot product of these user representations $(e_i \cdot e_j)$, this can lead to overestimating chat duration for users with similar profiles, resulting in sub-optimal recommendations when recommending similar users is not always ideal [28, 29]. To more accurately capture mutual interest while avoiding overestimation for similar users, we linearly project user representations into separate latent spaces:

$$\bar{e}_i = \mathbf{W}_1 e_i + \mathbf{b}_1, \quad \bar{e}_j = \mathbf{W}_2 e_j + \mathbf{b}_2, \quad (7)$$

where \mathbf{W}_1 and \mathbf{b}_1 are the learnable weight matrix and bias for the projection of the representation of user u_i , and \mathbf{W}_2 and \mathbf{b}_2 are the corresponding weight matrix and bias for their chat counterpart u_j . Then, the predicted chat duration is estimated by the dot product of the mapped representations \bar{e}_i and \bar{e}_j :

$$\hat{y}_{ij} = \bar{e}_i \cdot \bar{e}_j. \quad (8)$$

Exponential Transformation As plotted in Figure 4, the actual chat durations follow a long-tailed distribution in real-world social discovery platforms (blue histogram) in practice. However, when trained with naive *MSE* objective, predictions based on the dot product tend to follow a normal distribution (red histogram), which deviates from the true distribution.

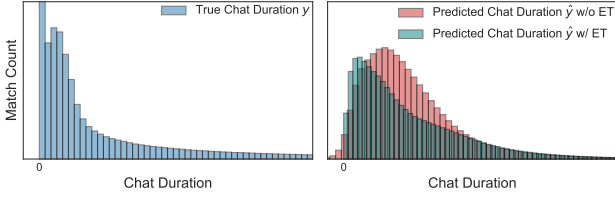


Fig. 4: (Left): True chat duration distribution. (Right): Predicted chat duration *with* and *without* exponential transform.

To match predictions with the true distribution, we apply an exponential transformation to Equation 8 as follows:

$$\hat{y}_{ij} = f_o(e_i, e_j) = \exp(w(\bar{e}_i \cdot \bar{e}_j) + b), \quad (9)$$

where w and b are learnable parameters.

As a result, the exponential transformation effectively adjusts the predicted durations to match the long-tailed distribution of actual chat durations (green histogram) with minimal computational overhead.

E. Two-Phase Training

Training session-based recommendation systems in real-time contexts poses significant computational challenges due to the dynamic nature of user representations. Each user’s preferences evolve after each interaction, requiring the system to frequently update their session representations to accurately reflect their current state. To make precise recommendations, the system must consider the updated session data for both users involved in each match. Traditionally, this involves processing and updating the session data for both the initiating user u_i and their chat counterpart u_j separately using complex models like transformers. For each user, the causal transformer processes their session history with a computational complexity of $O(|S|^2)$, where $|S|$ is the average session length per user. Modeling both users separately effectively doubles the computational cost, making it computationally intensive.

Moreover, accurately predicting matches requires jointly modeling how the sessions of u_i and u_j interact, which significantly increases computational overhead. This is because every interaction in u_i ’s session might influence and be influenced by every interaction in u_j ’s session, expanding the interaction space exponentially. In a naive approach, considering cross-attention between both users’ sequences could lead to a theoretical complexity of $O(|S|^4)$. Such high computational demands make real-time processing prohibitive, especially in large-scale platforms with millions of users and high interaction rates like *Azar*. The sequential dependencies inherent in causal transformers further exacerbate the issue, as each interaction’s representation depends on all previous interactions, leading to extensive computations. To address this challenge and improve training efficiency, we propose a *Two-Phase Training Strategy*, outlined in Algorithm 1, which significantly reduces computational overhead during training without substantially compromising the model’s performance.

Phase 1: Training Embedding Layers The primary goal of this phase is to efficiently train the user feature embedding

Algorithm 1 Two-Phase Training Strategy

- 1: **Input:** feature embedding layer f_u , auxiliary feature embedding layer \tilde{f}_u , session embedding layer f_s , and chat duration prediction layer f_o
 - 2: **Output:** the trained layers f_u , f_s , and f_o
 - 3: **# Phase 1 Training (Embedding Layer)**
 - 4: **repeat**
 - 5: **for** $u_i \in \mathcal{U}$ **do**
 - 6: compute $f_s(S_i) = [e_{i,0}^s, e_{i,1}^s, e_{i,2}^s, \dots, e_{i,h}^s]$
 - 7: **for** $m_k = (u_i, u_j, y_{ij,k}) \in S_i$ **do**
 - 8: compute $e_{i,k}^u = f_u(X_{i,k})$, $\tilde{e}_{j,k}^u = \tilde{f}_u(X_{j,k})$
 - 9: compute $\mathcal{L}_{MSE} = (f_o(e_{i,k}^u + e_{i,k-1}^s, \tilde{e}_{j,k}^u) - y_{ij,k})^2$
 - 10: update $f_u, \tilde{f}_u, f_s, f_o$ with \mathcal{L}_{MSE}
 - 11: **end for**
 - 12: **end for**
 - 13: **until** CUPID converges
 - 14: **# Phase 2 Training (Prediction Layer)**
 - 15: freeze the feature embedding layer f_u and session embedding layer f_s
 - 16: compute $e_i^u, e_i^s, e_j^u, e_j^s$ in advance
 - 17: **repeat**
 - 18: **for** $m = (u_i, u_j, y_{ij}) \in \mathcal{D}$ **do**
 - 19: compute $\mathcal{L}_{MSE} = (f_o(e_i^u + e_i^s, e_j^u + e_j^s) - y_{ij})^2$
 - 20: update f_o with \mathcal{L}_{MSE}
 - 21: **end for**
 - 22: **until** CUPID converges
-

layer f_u and the asynchronous session embedding layer f_s . We introduce an auxiliary user feature embedding layer \tilde{f}_u to assist in training these layers excluding session information from the chat counterparts. This reduces the input to $(X_i; S_i, X_j)$, allowing us to leverage the causal transformer to generate session representations $e_{i,k}^s$ with a single forward pass per user. This phase is aimed at minimizing the following objective:

$$\mathcal{L}_{MSE} = \frac{1}{|\mathcal{D}|} \sum_{u_i \in \mathcal{U}} \sum_{m_k \in S_i} (f_o(e_{i,k}^u + e_{i,k-1}^s, \tilde{e}_{j,k}^u) - y_{ij,k})^2, \quad (10)$$

where $e_{i,k-1}^s$ is the session state of user u_i after the $(k-1)$ -th match for predicting the chat duration of the k -th match.

Phase 2: Training the Chat Duration Prediction Layer In this phase, we enhance the chat duration prediction layer f_o by fully incorporating session information from both users in each match $(X_i; S_i, X_j; S_j)$. Thereby, the objective becomes:

$$\mathcal{L}_{MSE} = \frac{1}{|\mathcal{D}|} \sum_{u_i \in \mathcal{U}} \sum_{m_k \in S_i} (f_o(e_{i,k}^u + e_{i,k-1}^s, e_{j,k}^u + e_{j,k-1}^s) - y_{ij,k})^2. \quad (11)$$

In this final phase, we discontinue using the auxiliary user feature embedding layer \tilde{f}_u from the first phase and freeze the embedding layers (f_u, f_s) to improve processing efficiency. By computing the user feature representations in advance, subsequent calculations can be optimized.

Computational Complexity Analysis We analyze how our two-phase training strategy enhances training efficiency, as detailed in Table I. Let N denote the total number of training epochs in standard learning, with N_1 and N_2 for the first and second phases, respectively. $|\mathcal{D}|$ denotes the number of matching histories in the dataset, and $|S|$ indicates the average session length per user. In standard training, modeling session

TABLE I: Comparison of the number of causal transformer inferences with and without our two-phase training.

Phase		Time Complexity	
w/o Two-phase		$2N \mathcal{D} $	
w/ Two-phase	Phase-1	$N_1 \mathcal{D} / \mathcal{S} $	$(N_1 + 2) \mathcal{D} / \mathcal{S} $
	Phase-2	$2 \mathcal{D} / \mathcal{S} $	
Reduction Factor		$2N \mathcal{S} /(N_1 + 2)$	

representations for both users requires $2N|\mathcal{D}|$ inferences since both u_i and u_j need to be processed for each match history across all epochs. In contrast, the first phase of our method requires only $N_1|\mathcal{D}|/|\mathcal{S}|$ inferences, as we compute session representations for only one user per inference and leverage the average session length to reduce computations. During the second phase, by pre-extracting user representations and freezing the embedding layers f_s and f_u , the total number of inferences needed is just $2|\mathcal{D}|/|\mathcal{S}|$, regardless of N_2 . This method reduces the inferences required by the causal transformer in f_s to:

$$\frac{2N|\mathcal{S}|}{N_1 + 2}$$

compared to $2N|\mathcal{D}|$ in conventional methods. Assuming $N = N_1 = 10$ and $|\mathcal{S}| = 128$, our method achieves a **213x reduction** in transformer inferences. Considering that transformer inference constitutes the majority of training latency, this substantial reduction greatly facilitates the efficient training of CUPID, even with large datasets.

IV. EXPERIMENTS

A. Experimental Setups

Data Setups The performance of CUPID is evaluated in both offline and online environments. In the *offline* evaluation, its performance is tested in a controlled setting. A large-scale matching history from *Azar* is used, consisting of a billion-scale dataset from millions of user sessions generated over a month. Data from the last two days is used for validation and testing, while the remaining data is used as the training set. In the *online* evaluation, CUPID’s effectiveness is validated in real-world conditions to ensure that the gains observed are consistent in a live service environment.

Evaluation Setups We adopt two baseline models based on Wide&Deep [27], which were previously used in *Azar* before adopting session-based recommendations as follows:

- **Wide&Deep:** A widely adopted recommendation method that captures higher-order interactions among input features using neural networks. It employs user representations $e_i = e_i^u$ and $e_j = e_j^u$ are employed in Equation 6 without session representations e_i^s and e_j^s . Consequently, it relies solely on static user features and does not include any real-time information from user sessions.
- **Wide&Deep-S:** A variant of Wide&Deep that incorporates three real-time features generated during user sessions. It captures

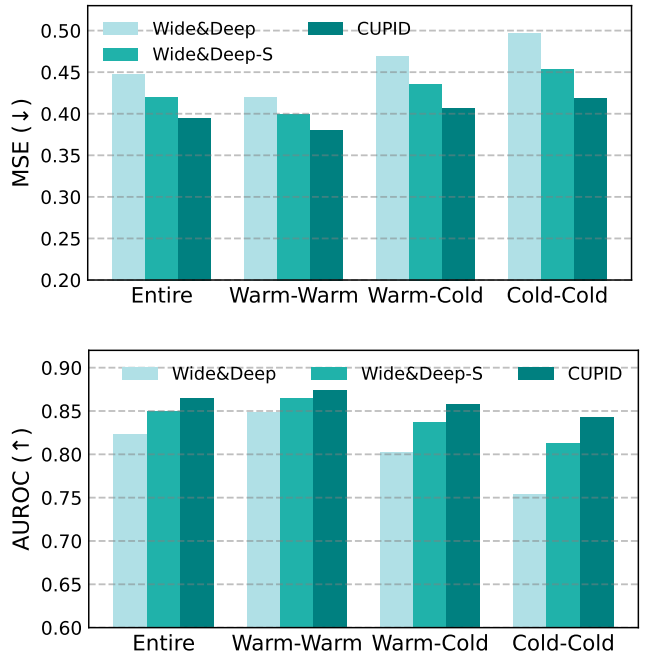


Fig. 5: *Offline* evaluation results on four types of matches.

TABLE II: *Online* evaluation results across all user segments in *Azar* production. The relative performance change of CUPID compared to the baseline Wide&Deep is reported.

User Segment	Average Chat Duration	Long Match Ratio	Short Match Ratio
All users	+6.8%	+12.6%	-2.4%
Warm-start users	+6.8%	+12.9%	-2.3%
Cold-start users	+5.9%	+9.7%	-4.1%

user behaviors while maintaining low latency by leveraging aggregated features from recent match histories, such as average chat duration, along with existing user features. This baseline serves to demonstrate the effectiveness of our sequential approach for modeling user sessions.

For performance evaluation, we use *MSE* and *Area Under the Receiver Operating Characteristic (AUROC)*. *MSE* measures the average squared difference between actual and predicted chat durations by applying log-scaled chat durations (ms) to minimize the impact of noise in shorter intervals. The same log-scaling is also used during the training of our recommendation models. In contrast, *AUROC* assesses the model’s ability to distinguish between potential matches that result in quality interactions and those that do not. A quality match is defined as one where the chat duration exceeds a specific threshold. As latency is another critical factor for real-time reciprocal recommendation, we also evaluate the latency improvement achieved by adopting CUPID in the real-world deployment of *Azar*.

B. Offline Performance Evaluation

In Figure 5, the overall performances across three match types are presented. The match types include *Entire Match*,

TABLE III: The simulation of the deployment environment where user session representations may not up-to-date. We observe the performance changes by using delayed user session representations stored at time $(t - t')$, with varying delay time t' .

Method	t'	Entire Match		Warm-Warm Match		Warm-Cold Match		Cold-Cold Match	
		MSE (\downarrow)	AUROC (\uparrow)	MSE (\downarrow)	AUROC (\uparrow)	MSE (\downarrow)	AUROC (\uparrow)	MSE (\downarrow)	AUROC (\uparrow)
CUPID	-	0.3968	0.8635	0.3815	0.8735	0.4094	0.8564	0.4214	0.8409
	2000ms	0.3989	0.8616	0.3834	0.8719	0.4117	0.8541	0.4239	0.8389
	4000ms	0.3990	0.8616	0.3834	0.8719	0.4118	0.8540	0.4239	0.8389
	8000ms	0.3993	0.8614	0.3837	0.8717	0.4121	0.8538	0.4242	0.8386
	16000ms	0.4004	0.8605	0.3848	0.8710	0.4132	0.8528	0.4254	0.8375
Wide&Deep-S	-	0.4197	0.8497	0.3996	0.8655	0.4359	0.8375	0.4539	0.8136

TABLE IV: Ablation test results. SP and ET denote the second phase in our two-phase learning and the exponential transform in Equation 9, respectively. The performance changes are observed by removing each component.

Components			Entire Match		Warm-Warm Match		Warm-Cold Match		Cold-Cold Match	
e^s	SP	ET	MSE (\downarrow)	AUROC (\uparrow)	MSE (\downarrow)	AUROC (\uparrow)	MSE (\downarrow)	AUROC (\uparrow)	MSE (\downarrow)	AUROC (\uparrow)
\times	\checkmark	\checkmark	0.4197	0.8497	0.3996	0.8655	0.4359	0.8375	0.4539	0.8136
\checkmark	\times	\checkmark	0.4271	0.8464	0.4059	0.8649	0.4436	0.8329	0.4648	0.7995
\checkmark	\checkmark	\times	0.3996	0.8615	0.3845	0.8712	0.4121	0.8545	0.4239	0.8394
\checkmark	\checkmark	\checkmark	0.3948	0.8648	0.3797	0.8745	0.4072	0.8577	0.4190	0.8431

TABLE V: Response latency of CUPID in online environments under the *Azar* service scenario.

Components	90-th percentile	99-th percentile
User representation e^u	9ms	17ms
Session representation e^s	236ms	290ms
Synchronous implementation	236ms	290ms
CUPID:Asynchronous implementation (Ours.)	48ms (-79.7%)	70ms (-75.9%)

which encompasses all categories; *Warm-Warm*, for matches between warm-start users; *Warm-Cold*, for matches between warm-start and cold-start users; and *Cold-Cold*, for matches exclusively between cold-start users. Here, cold-start users have no previous matching history in the training dataset. The distribution is 58.1% for *Warm-Warm*, 35.5% for *Warm-Cold*, and 6.3% for *Cold-Cold*. CUPID consistently outperforms baseline methods across all categories and metrics.

C. Online Production Performance

While CUPID shows a significant improvement in predicting satisfaction scores in offline experiments, it may not always lead to increased user engagement online. To evaluate its real-world impact, we test CUPID in the production environment of *Azar*, comparing it with the baselines. We conduct a Switchback [30] test instead of an A/B test due to the shared matching pool, which makes it difficult to independently separate A/B groups. The results in Table II show improvements in metrics such as average chat duration and the ratio of long to short matches, defined by a preset threshold. For all user segments, CUPID consistently increases the average chat

duration and improves match quality. This demonstrates that CUPID not only accurately predicts satisfaction scores but also enhances user experience in a live setting. Meanwhile, Table V shows the latency improvement achieved by CUPID, emphasizing its primary goal of delivering low-latency recommendations through asynchronous session modeling. In the real-world deployment of *Azar*, CUPID reduces latencies at the 90th and 99th percentiles by up to 79.7% compared to synchronous computation of session representations in the matching pipeline. This significant reduction ensures stable latency, which is essential for real-time services.

D. Effect of Delayed Session Representation

In real deployment, the session representation e^s might miss the latest matching histories if a user requests a new match before the update is complete. The system then uses a delayed representation, which lacks data from the most recent matches. To study the impact of this delay, we simulate an environment where the representation update is delayed for t' milliseconds and predict chat durations for users in the matching pool $\mathcal{U}^{(t)}$ using this delayed data. The results are summarized in Table III. Two main observations emerge. First, prediction performance slightly decreases as delay time increases, which is expected since the system design decouples session modeling from the synchronous matching pipeline to avoid latency issues. This compromise is acceptable, as it prevents session modeling from becoming a bottleneck. Second, even with this delay, the models still outperform the Wide&Deep-S baseline by a significant margin in all cases while maintaining similar latency. This shows that the approach, with its decoupled session modeling, achieves an optimal balance between latency and prediction performance.

E. Ablation Study

An ablation test is conducted to evaluate the impact of individual components on CUPID’s performance, focusing on session representation e^s , the Exponential Transformation (ET), and the second phase of the two-phase training strategy. The results, shown in Table IV, indicate a performance drop when any component is removed. Excluding the session representation results in a significant decline, especially for cold-start users, underscoring its role in capturing mutual interests. Skipping the second-phase training also negatively impacts performance, highlighting its importance in using session data from both users to improve chat duration predictions. Additionally, omitting the exponential transformation leads to poorer performance, underscoring its value in aligning predicted chat durations with the actual distribution and stabilizing model training.

V. RELATED WORK

Reciprocal Recommendation Reciprocal recommendation systems differ from conventional in the sense of they aim to enhance mutual satisfaction through user-to-user recommendations [4, 31, 32]. These systems have been widely studied, especially in contexts like online dating [33–36], and job search platforms [29, 37–39]. Our work shifts the focus to real-time reciprocal recommendations, where candidates appear and disappear dynamically. This is the first comprehensive study to investigate these complex dynamics in real-time.

Session-Based Recommendation Session-based recommendation systems predict the next item by capturing dynamic user behaviors and intents within a session. Various models, such as Markov Chains [40, 41], recurrent neural networks [22, 42, 43], graph neural networks [44–47], transformers [20, 48–51], and other attention mechanisms [52–54] have been utilized for this purpose. Our study extends session-based recommendations into the underexplored area of reciprocal recommendation tasks. While [55] examines sequential recommendations in a two-sided market, it does not address the low-latency requirements essential for real-time one-on-one social discovery platforms. In contrast, our work specifically focuses on meeting these extreme low-latency constraints, facilitating rapid and efficient user matching in reciprocal session-based recommendation systems.

VI. CONCLUSION

To the best of our knowledge, this is the first study to develop a session-based reciprocal recommendation system optimized for real-time social discovery platforms. Our approach tackles stringent latency requirements by using asynchronous session modeling, which significantly reduces the time required for processing. Additionally, we introduce an efficient two-phase training method that simplifies the complexities of combining session-based and reciprocal recommendations. Our system, validated on a large-scale offline dataset and in a real-world environment, increases average

chat duration by 6.8% for warm-start users and 5.9% for cold-start users. Moreover, it achieves over a 76% reduction in latency compared to purely synchronous methods. This research opens a new direction for session-based real-time reciprocal recommendations.

Ethical Statement By introducing CUPID, we aim to enhance user engagement and satisfaction through efficient, personalized matchmaking in social discovery. Using asynchronous session modeling and a two-phase training strategy, CUPID addresses low latency and dynamic user preferences. However, deploying such a system involves ethical considerations, including user privacy, data security, and potential algorithmic biases. To address these, we ensure strict adherence to data protection laws, implement robust security measures, and commit to developing fairness-aware algorithms with regular audits to prevent unintended discrimination.

REFERENCES

- [1] Z. Zheng, X. Hu, S. Gao, H. Zhu, and H. Xiong, “Mirror: A multi-view reciprocal recommender system for online recruitment,” in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 543–552.
- [2] B. A. Potts, H. Khosravi, C. Reidsema, A. Bakharia, M. Belonogoff, and M. Fleming, “Reciprocal peer recommendation for learning purposes,” in *Proceedings of the 8th international conference on learning analytics and knowledge*, 2018, pp. 226–235.
- [3] Y. Zheng, T. Dave, N. Mishra, and H. Kumar, “Fairness in reciprocal recommendations: A speed-dating study,” in *Adjunct publication of the 26th conference on user modeling, adaptation and personalization*, 2018, pp. 29–34.
- [4] I. Palomares, C. Porcel, L. Pizzato, I. Guy, and E. Herrera-Viedma, “Reciprocal recommender systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation,” *Information Fusion*, vol. 69, pp. 103–127, 2021.
- [5] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay, “Recon: a reciprocal recommender for online dating,” in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 207–214.
- [6] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian, “A survey on session-based recommender systems,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–38, 2021.
- [7] S. Wang, Q. Zhang, L. Hu, X. Zhang, Y. Wang, and C. Aggarwal, “Sequential/session-based recommendations: Challenges, approaches, applications and opportunities,” in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 3425–3428.
- [8] M. Ludewig, N. Mauro, S. Latifi, and D. Jannach, “Performance comparison of neural and non-neural approaches to session-based recommendation,” in *Proceed-*

- ings of the 13th ACM conference on recommender systems, 2019, pp. 462–466.
- [9] Y. K. Tan, X. Xu, and Y. Liu, “Improved recurrent neural networks for session-based recommendations,” in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 17–22.
- [10] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, “Personalizing session-based recommendations with hierarchical recurrent neural networks,” in *proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 130–137.
- [11] Z. Liu, L. Zou, X. Zou, C. Wang, B. Zhang, D. Tang, B. Zhu, Y. Zhu, P. Wu, K. Wang *et al.*, “Monolith: real time recommendation system with collisionless embedding table,” *arXiv preprint arXiv:2209.07663*, 2022.
- [12] Z. Hou, F. Bu, Y. Zhou, L. Bu, Q. Ma, Y. Wang, H. Zhai, and Z. Han, “Dycars: A dynamic context-aware recommendation system,” *Mathematical Biosciences and Engineering*, vol. 21, no. 3, pp. 3563–3593, 2024.
- [13] A. Mahyari, P. Pirolli, and J. A. LeBlanc, “Real-time learning from an expert in deep recommendation systems with application to mhealth for physical exercises,” *IEEE journal of biomedical and health informatics*, vol. 26, no. 8, pp. 4281–4290, 2022.
- [14] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for cold-start recommendations,” in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002, pp. 253–260.
- [15] R. Sethi and M. Mehrotra, “Cold start in recommender systems—a survey from domain perspective,” in *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020*. Springer, 2021, pp. 223–232.
- [16] D. K. Panda and S. Ray, “Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review,” *Journal of Intelligent Information Systems*, vol. 59, no. 2, pp. 341–366, 2022.
- [17] N. A. Abdullah, R. A. Rasheed, M. H. N. M. Nasir, and M. M. Rahman, “Eliciting auxiliary information for cold start user recommendation: A survey,” *Applied Sciences*, vol. 11, no. 20, p. 9608, 2021.
- [18] F. Berisha and E. Bytyçi, “Addressing cold start in recommender systems with neural networks: a literature survey,” *International Journal of Computers and Applications*, vol. 45, no. 7-8, pp. 485–496, 2023.
- [19] X. Zheng, R. Wu, Z. Han, C. Chen, L. Chen, and B. Han, “Heterogeneous information crossing on graphs for session-based recommender systems,” *ACM Transactions on the Web*, vol. 18, no. 2, pp. 1–24, 2024.
- [20] G. de Souza Pereira Moreira, S. Rabhi, J. M. Lee, R. Ak, and E. Oldridge, “Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation,” in *Proceedings of the 15th ACM Conference on Recommender Systems*, 2021, pp. 143–153.
- [21] J. Wang, K. Ding, Z. Zhu, and J. Caverlee, “Session-based recommendation with hypergraph attention networks,” in *Proceedings of the 2021 SIAM international conference on data mining (SDM)*. SIAM, 2021, pp. 82–90.
- [22] S. Liu and Y. Zheng, “Long-tail session-based recommendation,” in *Proceedings of the 14th ACM Conference on Recommender Systems*, 2020, pp. 509–514.
- [23] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, “Contextual and sequential user embeddings for large-scale music recommendation,” in *Proceedings of the 14th ACM Conference on Recommender Systems*, 2020, pp. 53–62.
- [24] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. S. Kim, P. Compton, and A. Mahidadia, “Reciprocal and heterogeneous link prediction in social networks,” in *Advances in Knowledge Discovery and Data Mining: 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29–June 1, 2012, Proceedings, Part II 16*. Springer, 2012, pp. 193–204.
- [25] P. Xia, B. Liu, Y. Sun, and C. Chen, “Reciprocal recommendation system for online dating,” in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015, pp. 234–241.
- [26] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry *et al.*, “An updated set of basic linear algebra subprograms (blas),” *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.
- [27] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, “Wide & deep learning for recommender systems,” in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.
- [28] J. Neve and R. McConville, “Imrec: Learning reciprocal preferences using images,” in *Proceedings of the 14th ACM Conference on Recommender Systems*, 2020, pp. 170–179.
- [29] C. Yang, Y. Hou, Y. Song, T. Zhang, J.-R. Wen, and W. X. Zhao, “Modeling two-way selection preference for person-job fit,” in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 102–112.
- [30] J. Robins, “A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect,” *Mathematical modelling*, vol. 7, no. 9-12, pp. 1393–1512, 1986.
- [31] H. Abdollahpouri, G. Adomavicius, R. Burke, I. Guy, D. Jannach, T. Kamishima, J. Krasnodebski, and L. Pizzato, “Multistakeholder recommendation: Survey and research directions,” *User Modeling and User-Adapted Interaction*, vol. 30, pp. 127–158, 2020.
- [32] I. Palomares, “Reciprocal recommendation: Matching users with the right users,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2429–

- [33] J. Neve and I. Palomares, “Latent factor models and aggregation operators for collaborative filtering in reciprocal recommender systems,” in *Proceedings of the 13th ACM conference on recommender systems*, 2019, pp. 219–227.
- [34] Y. Tomita, R. Togashi, and D. Moriwaki, “Matching theory-based recommender systems in online dating,” in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 538–541.
- [35] A. Alanazi and M. Bain, “A people-to-people content-based reciprocal recommender using hidden markov models,” in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 303–306.
- [36] K. Tu, B. Ribeiro, D. Jensen, D. Towsley, B. Liu, H. Jiang, and X. Wang, “Online dating recommendations: matching markets and learning preferences,” in *Proceedings of the 23rd international conference on world wide web*, 2014, pp. 787–792.
- [37] J. Jiang, S. Ye, W. Wang, J. Xu, and X. Luo, “Learning effective representations for person-job fit by feature fusion,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2549–2556.
- [38] Y. Lu, S. El Helou, and D. Gillet, “A recommender system for job seeking and recruiting website,” in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 963–966.
- [39] R. Yan, R. Le, Y. Song, T. Zhang, X. Zhang, and D. Zhao, “Interview choice reveals your preference on the market: To improve job-resume matching through profiling memories,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 914–922.
- [40] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, “Factorizing personalized markov chains for next-basket recommendation,” in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 811–820.
- [41] R. He and J. McAuley, “Fusing similarity models with markov chains for sparse sequential recommendation,” in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 191–200.
- [42] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks,” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: <http://arxiv.org/abs/1511.06939>
- [43] D. Jannach and M. Ludewig, “When recurrent neural networks meet the neighborhood for session-based recommendation,” in *Proceedings of the eleventh ACM conference on recommender systems*, 2017, pp. 306–310.
- [44] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, “Session-based recommendation with graph neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 346–353.
- [45] J. Guo, Y. Yang, X. Song, Y. Zhang, Y. Wang, J. Bai, and Y. Zhang, “Learning multi-granularity consecutive user intent unit for session-based recommendation,” in *Proceedings of the fifteenth ACM International conference on web search and data mining*, 2022, pp. 343–352.
- [46] P. Zhang, J. Guo, C. Li, Y. Xie, J. B. Kim, Y. Zhang, X. Xie, H. Wang, and S. Kim, “Efficiently leveraging multi-level user intent for session-based recommendation via atten-mixer network,” in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 168–176.
- [47] Y. Pang, L. Wu, Q. Shen, Y. Zhang, Z. Wei, F. Xu, E. Chang, B. Long, and J. Pei, “Heterogeneous global graph neural networks for personalized session-based recommendation,” in *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022, pp. 775–783.
- [48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [49] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, “Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer,” in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.
- [50] X. Xia, J. Yu, Q. Wang, C. Yang, N. Q. V. Hung, and H. Yin, “Efficient on-device session-based recommendation,” *ACM Transactions on Information Systems*, vol. 41, no. 4, pp. 1–24, 2023.
- [51] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, “S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization,” in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 1893–1902.
- [52] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural attentive session-based recommendation,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1419–1428.
- [53] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, “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*, 2018, pp. 1831–1839.
- [54] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, “Filter-enhanced mlp is all you need for sequential recommendation,” in *Proceedings of the ACM web conference 2022*, 2022, pp. 2388–2399.
- [55] B. Zheng, Y. Hou, W. X. Zhao, Y. Song, and H. Zhu, “Reciprocal sequential recommendation,” in *Proceedings of the 17th ACM Conference on Recommender Systems*, 2023, pp. 89–100.