Lightweight Yet Fine-Grained: A Graph Capsule Convolutional Network with Subspace Alignment for Shared-Account Sequential Recommendation

Jinyu Zhang[1], **Zhongying Zhao**[1*], **Chao Li**[1], **Yanwei Yu**[2]

[1]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]College of Computer Science and Technology, Ocean University of China, Qingdao 266400, China
jinyuz1996@outlook.com, zzysuin@163.com or zyzhao@sdust.edu.cn, lichao@sdust.edu.cn, yuyanwei@ouc.edu.cn

Abstract

Shared-account Sequential Recommendation (SSR) aims to provide personalized recommendations for accounts shared by multiple users with varying sequential preferences. Previous studies on SSR struggle to capture the fine-grained associations between interactions and different latent users within the shared account's hybrid sequences. Moreover, most existing SSR methods (e.g., RNN-based or GCN-based methods) have quadratic computational complexities, hindering the deployment of SSRs on resource-constrained devices. To this end, we propose a **Light**weight **G**raph **C**apsule **C**onvolutional **N**etwork with subspace alignment for shared-account sequential recommendation, named LightGC^{2}N. Specifically, we devise a lightweight graph capsule convolutional network. It facilitates the fine-grained matching between interactions and latent users by attentively propagating messages on the capsule graphs. Besides, we present an efficient subspace alignment method. This method refines the sequence representations and then aligns them with the finely clustered preferences of latent users. The experimental results on four real-world datasets indicate that LightGC^{2}N outperforms nine state-of-the-art methods in accuracy and efficiency.

Code —
https://github.com/ZZY-GraphMiningLab/LightGC2N

Introduction

Sequential Recommender systems (SRs) strive to provide users with personalized content (Ma et al. 2024), products (Yue et al. 2023), or services (Li et al. 2024) based on their sequential preferences. Most SRs are under an ideal assumption that each account is merely associated with one single user (Verstrepen and Goethals 2015). In real-world scenarios, many users prefer to share their accounts with family members or close friends (Jiang et al. 2018). As illustrated in Figure 1, multiple family members (i.e., latent users) utilize a shared video account. Clearly, their viewing histories reveal distinct preferences, yet they're blended into an account-level hybrid sequence. Distinguishing the diverse preferences of latent users while providing account-level sequential recommendations emerges as an appealing yet challenging task, i.e., the Shared-account Sequential Recommendation (SSR).

Figure 1: An example to illustrate the shared-account sequential recommendation scenario. $\{v_1, v_2, \ldots, v_6\}$ are the historical behaviors in the hybrid sequence.

In the early explorations of the SSR, researchers prefer utilizing Recurrent Neural Networks (RNNs) to distinguish the user preferences from shared accounts (Wen et al. 2021). π-Net (Ma et al. 2019) and PSJNet (Sun et al. 2023) are two RNN-based SSR methods that leverage Gating Recurrent Unit (GRU) to cluster the preferences of latent users and then learn the account-level sequence representations. Since an account encompasses multiple latent users, the shared-account sequences tend to be longer than typical sequences. However, RNN-based methods have catastrophic forgetting problems in processing lengthy sequences, leading to gradient vanishing problems during model training (Guo et al. 2024). Subsequently, DA-GCN (Guo et al. 2021) and TiDA-GCN (Guo et al. 2024) are proposed. These graph-based SSR methods model the preferences of each latent user through a multi-head self-attention mechanism and learn account-level sequence embeddings by propagating messages on the sequential graph. Although these methods have achieved remarkable performance in SSR, they still face the below two challenges:

(1) Coarse-grained user representations. Since an account is shared by multiple users in SSR, the interactions are generated by diverse latent users. Most graph-based SSR methods directly learn the preferences of latent users by modeling the account-level sequences on the sequential graphs (Wen et al. 2021). They try to perform clustering (Guo et al. 2021) or apply multi-head self-attention (Guo

*Corresponding Author.

et al. 2024) to extract the multiple interests of each account, thereby simulating the diverse preferences of latent users. However, these methods fail to distinguish the ownership of each interaction within the hybrid sequences, making it challenging for SSRs to capture the fine-grained preferences of latent users.

(2) High computational complexity. The graph-based shared-account sequential recommenders usually integrate complicated structures (e.g., self-attention (Guo et al. 2024) or clustering methods (Sun et al. 2023; Jiang et al. 2018)) to differentiate the preferences of multiple latent users from hybrid sequences. However, both the self-attention mechanism (Katharopoulos et al. 2020) and the clustering method based on self-representation matrices (Cai et al. 2022) have quadratic computational complexity. Such a high computational demand seriously impacts the user experience and poses a challenge for deploying graph-based SSRs on resource-constrained mobile devices.

To tackle the above problems, we propose a **Light**weight **G**raph **C**apsule **C**onvolutional **N**etwork (LightGC^{2}N). Specifically, we present a lightweight Graph Capsule Convolutional Network (GC^{2}N) to identify the ownership of each interaction for latent users. In this component, we construct capsule graphs to identify the ownership of interactions for different latent users. By attentively propagating messages on the graphs, GC^{2}N performs a fine-grained distinction of the preferences for different latent users. Furthermore, we design an account-level dynamic routing mechanism. It merges the preferences of latent users, yielding the account-level capsule representations. Besides, we devise an efficient Subspace Alignment (SA) method that utilizes low-rank subspace bases to refine the sequence embeddings. Finally, SA adopts a contrastive learning strategy to align the preferences of latent users between refined and original sequences.

The main contributions of this work includes:

- We propose a subspace alignment-enhanced graph capsule convolutional network for the shared-account sequential recommendation, namely LightGC^{2}N.
- We design a lightweight graph capsule convolutional network that finely distinguishes the preferences of each latent user within an account.
- We devise an efficient subspace alignment method that refines the sequence embeddings and aligns them with the preferences of latent users.
- Experimental results on four datasets demonstrate that LightGC^{2}N outperforms other state-of-the-art SSR methods in terms of performance and model efficiency.

Related Work

Sequential Recommendation

Sequential recommender systems (SRs) predict users' next interactions based on their sequential preferences (Chen et al. 2024). Early researches utilized Markov chains to address the sparsity issues in SR tasks (He and McAuley 2016; Cai, He, and McAuley 2017), but they fail to capture the dynamics of user preferences. Subsequently, researchers begin to explore deep neural networks for SRs, including RNN-based methods (Quadrana et al. 2017), GNN-based methods (Fan et al. 2021), attention-based methods (Kang and McAuley 2018; Shin et al. 2024; He et al. 2018), and contrastive learning-based methods (Xie et al. 2022a). These deep learning-based SR methods excel in capturing dynamic sequential patterns of users. Nevertheless, they typically assume that each account is associated with a single user (Guo et al. 2023), thus failing to provide accurate recommendations for shared accounts.

Shared-account Sequential Recommendation

Shared-account Sequential Recommender systems (SSRs) aim to identify the diverse preferences of latent users while providing personalized recommendations for shared accounts (Verstrepen and Goethals 2015). Early SSR research utilized RNN-based methods (Ma et al. 2019; Sun et al. 2023) to identify latent users, employing GRU units to filter out information from each account. However, these methods suffer from gradient vanishing issues with long sequences. Subsequently, the graph-based SSR methods with attention networks are proposed (Guo et al. 2021, 2024), which propagates user-specific messages to identify latent users. These graph-based methods have achieved remarkable performance on SSR. However, they didn't consider the fine-grained associations between interactions in sequences and latent users. Besides, their high computational complexity hinders the deployment of SSRs on resource-constrained edge devices (e.g., smartphones or tablets).

Methodology

Preliminaries

Notations. Suppose that $\mathcal{I} = \{I_1, I_2, \ldots, I_t, \ldots, I_m\}$ is the set of items, where I_t denotes the t-th item. The set of shared accounts is denoted as $\mathcal{A} = \{A_1, A_2, \ldots, A_k, \ldots, A_n\}$, where A_k represents the k-th shared account. Moreover, the set of sequences is denoted as $\mathcal{S} = \{S_1, S_2, \ldots, S_k, \ldots, S_n\}$, where S_k denotes the hybrid sequence of the shared account A_k. Suppose that each account contains α latent users, e.g., $A_k = \{u_{k,1}, \ldots, u_{k,h}, \ldots, u_{k,\alpha}\}$, where $u_{k,h}$ denotes the h-th latent user in the account A_k.

Problem Definition. Given S_k and A_k, the task of SSR is to recommend the next item I_{t+1} that A_k is most likely to consume, based on the account's hybrid sequence S_k. The probabilities of all recommendation candidates are represented as:

$$P(I_{t+1}|S_k, A_k) \sim f(S_k, A_k), \tag{1}$$

where $P(I_{t+1}|S_k, A_k)$ denotes the probability of recommending I_{t+1} to A_k given its historical hybrid sequence S_k, and $f(S_k, A_k)$ is the function designed to estimate the probability.

Framework of LightGC^{2}N

As shown in Figure 2, the LightGC^{2}N consists of four key components: 1) sequential graph construction, 2) graph cap-

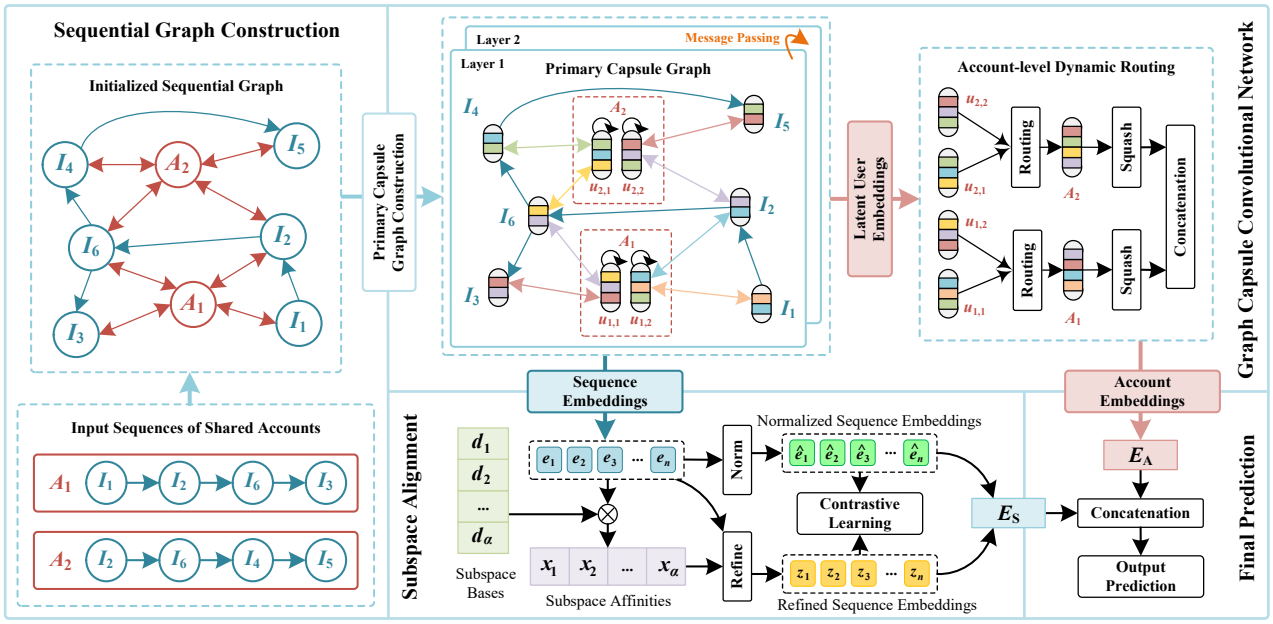

Figure 2: Framework of LightGC^{2}N, where A_1 and A_2 represent two shared accounts, and $\{I_1, I_2, \ldots, I_6\}$ denote the historical interactions that compose the hybrid sequences for these accounts.

sule convolutional network, 3) subspace alignment, and 4) final prediction. The details are given below.

Sequential Graph Construction

The sequential graphs are constructed as the input for the graph capsule convolutional network. During the graph construction, two types of associations are considered: account-item interactive relations and sequential dependencies between items. We define the sequential graphs as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges. Each edge in $\mathcal{E}$ denotes a relation between nodes. Specifically, the adjacency matrix $\mathcal{M} \in \mathbb{R}^{(m+n)\times(m+n)}$ of the sequential graph $\mathcal{G}$ is denoted by Eqn. (2).

$$\mathcal{M} = \begin{bmatrix} \mathbf{M}_S & \mathbf{M}_I \\ \mathbf{M}_I^\top & \mathbf{0} \end{bmatrix}, \tag{2}$$

where $\mathbf{M}_S \in \mathbb{R}^{m\times m}$ denotes the adjacency matrix carrying the sequential relationships between items, and each entry $M_{ij} = 1$ if item I_j is the prior of item I_i in the input sequences; $M_{ij} = 0$ otherwise. $\mathbf{M}_I \in \mathbb{R}^{m\times n}$ represents the adjacency matrix containing the interactive relationships between accounts and items, each entry $M_{kl} = 1$ if account A_k has interaction with item I_l; $M_{kl} = 0$ otherwise.

Graph Capsule Convolutional Network (GC^{2}N)

We design the GC^{2}N is to capture the fine-grained preferences of each latent user within a shared account. The inputs to GC^{2}N are the associations among nodes from the initialized sequential graphs and the embeddings at the 0-th layer, i.e., $\mathbf{E}_I^{(0)} \in \mathbb{R}^{m\times d_1}$ for all items and $\mathbf{E}_A^{(0)} \in \mathbb{R}^{n\times d_1}$ for all accounts.

Primary Capsule Graph Construction. In this component, node embeddings are projected into high-dimensional capsule spaces to construct the primary capsule graphs, thereby exploring the fine-grained differences in preferences between latent users. Specifically, this component utilizes linear attention mechanism (Katharopoulos et al. 2020) to project item embeddings $\mathbf{E}_I^{(0)}$ into item capsules $\mathbf{C}_I^{(0)} \in \mathbb{R}^{m\times d_2}$, where the $\mathbf{E}_I^{(0)}$ is treated as **Q**uery, **K**ey and **V**alue during the calculation. In contrast to the self-attention mechanism, the linear attention mechanism first calculates the outer product between the **K**ey matrix and the **V**alue matrix to obtain the attention map, and then calculates the correlations between the map and the **Q**uery matrix. Since the linear attention changes the calculation orders, it achieves lower computational complexity (i.e., $\mathcal{O}(N \times d^2)$) while capturing global associations among items. The calculation of the linear attention mechanism is formulated as Eqn. (3).

$$\mathbf{C}_I^{(0)} = \mathbf{Q}\left(\text{softmax}\left(\frac{\mathbf{K}^\top\mathbf{V}}{\sqrt{d_1}}\right)\right)\mathbf{W}_l + \mathbf{b}_l, \tag{3}$$

where $\mathbf{W}_l \in \mathbb{R}^{d_1\times d_2}$ is a dimension transformation matrix, and $\mathbf{b}_l \in \mathbb{R}^{1\times d_2}$ is the bias term.

Since there are no sequential relationships between accounts, it does not need to use an attention-based method for account representations $\mathbf{E}_A^{(0)}$. Instead, GC^{2}N leverages point-wise Conv1D to project them into capsule-form $\mathbf{C}_A^{(0)} \in \mathbb{R}^{n\times\alpha\times d_2}$. In the point-wise Conv1D, both the kernel size and stride are set to 1 for linear dimensional transformation (Wu et al. 2023). The calculation is denoted as Eqn. (4).

$$\mathbf{C}_A^{(0)} = \mathbf{E}_A^{(0)} * \mathbf{W}_c + \mathbf{b}_c, \tag{4}$$

where $*$ represents the convolutional operation, $\mathbf{W}_c \in \mathbb{R}^{d_1 \times \alpha \times d_2}$ denotes the kernel of Conv1D, $\mathbf{b}_c \in \mathbb{R}^{1 \times \alpha \times d_2}$ represents the bias, and α is a hyper-parameter that controls the number of latent users within shared accounts.

Subsequently, each account capsule $\mathbf{C}_{A_k}^{(0)} \in \mathbb{R}^{\alpha \times d_2}$ is split into α latent user capsules $\mathbf{C}_{u_k}^{(0)} \in \mathbb{R}^{d_2}$. Then, the model is able to match the interactions to the specific latent users within an account by attentively calculating the correlations between their capsules, which facilitates the fine-grained distinction of the preferences for latent users.

Fine-grained Message Propagation on capsule graphs. Taking the shared-account A_k as an example. The capsule embedding of the h-th latent user in A_k is denoted as $\mathbf{C}_{u_{k,h}}$. To realize fine-grained message passing on the primary capsule graphs, GC^{2}N calculates the correlations a_{I_j} between items $\mathbf{C}_{I_j}$ and user $\mathbf{C}_{u_{k,h}}$ as:

$$a_{I_j} = \frac{\exp\left(\mathbf{C}_{u_{k,h}} \cdot \mathbf{C}_{I_j}\right)}{\sum_{I_j \in \mathcal{N}_I^{A_k}} \sqrt{\exp\left(\mathbf{C}_{u_{k,h}} \cdot \mathbf{C}_{I_j}\right)}}, \tag{5}$$

where $\mathcal{N}_I^{A_k}$ is the set of all the interacted items of A_k.

Such an attentive calculation facilitates the fine-grained matching between interactions and latent users. Then, the messages propagated to $u_{k,h}$ at the l-th layer are denoted as:

$$\mathbf{m}_{u_{k,h} \leftarrow I_j}^{(l)} = \mathbf{W}_1^{(l)} \mathbf{C}_{I_j}^{(l-1)} + \mathbf{W}_2^{(l)} (a_{I_j}^{(l-1)} \mathbf{C}_{I_j}^{(l-1)}), \tag{6}$$

where $\mathbf{m}_{u_{k,h} \leftarrow I_j}^{(l)}$ denotes the passed message, $\mathbf{W}_1^{(l)}$ denotes the learnable weights that controls how much information should be passed from neighboring item I_j, and $\mathbf{W}_2^{(l)}$ is another learnable weighting matrix that controls the participation of correlations.

We also add self-connections to retain the independent characteristics of $\mathbf{C}_{u_{k,h}}^{(l)}$, which is formulated as:

$$\mathbf{m}_{u_{k,h} \leftarrow u_{k,h}}^{(l)} = \mathbf{W}_3^{(l)} \mathbf{C}_{u_{k,h}}^{(l-1)}, \tag{7}$$

where $\mathbf{m}_{u_{k,h} \leftarrow u_{k,h}}^{(l)}$ is the retained information of the user capsule from $(l-1)$-th layer, $\mathbf{W}_3^{(l)}$ is the learnable parameter that controls how much information of $\mathbf{C}_{u_{k,h}}^{(l-1)}$ should be retained.

Hence, the capsule representation $\hat{\mathbf{C}}_{u_{k,h}}^{(l)}$ of latent user $u_{k,h}$ is updated by Eqn. (8).

$$\hat{\mathbf{C}}_{u_{k,h}}^{(l)} = \sum_{I_j \in \mathcal{N}_I^{u_{k,h}}} \mathbf{m}_{u_{k,h} \leftarrow I_j}^{(l)} + \mathbf{m}_{u_{k,h} \leftarrow u_{k,h}}^{(l)}, \tag{8}$$

where $\mathcal{N}_I^{u_{k,h}}$ denotes the set of all items interacted by $u_{k,h}$.

Similarly, the message propagated to item capsule $\mathbf{C}_{I_j}$ at l-th layer is represented by Eqn. (9):

$$\mathbf{m}_{I_j \leftarrow u_{k,g}}^{(l)} = \mathbf{W}_4^{(l)} \mathbf{C}_{u_{k,g}}^{(l-1)}; \quad \mathbf{m}_{I_j \leftarrow I_{j-1}}^{(l)} = \mathbf{W}_5^{(l)} \mathbf{C}_{I_{j-1}}^{(l-1)}, \tag{9}$$

where $u_{k,g}$ represents a latent user who has interacted with I_j, $\mathbf{m}_{I_j \leftarrow u_{k,g}}^{(l)}$ is the message passed from $u_{k,g}$ to I_j, and I_{j-1} is the neighboring item of I_j, $\mathbf{m}_{I_j \leftarrow I_{j-1}}^{(l)}$ denotes the message passed from I_{j-1} to I_j, $\mathbf{W}_4^{(l)}$ and $\mathbf{W}_5^{(l)}$ are learnable weights.

Then, the capsule representation $\hat{\mathbf{C}}_{I_j}^{(l)}$ of item I_j is updated by Eqn. (10).

$$\hat{\mathbf{C}}_{I_j}^{(l)} = \sum_{u_{k,g} \in \mathcal{N}_u^{I_j}} \mathbf{m}_{I_j \leftarrow u_{k,g}}^{(l)} + \sum_{I_{j-1} \in \mathcal{N}_I^{I_j}} \mathbf{m}_{I_j \leftarrow I_{j-1}}^{(l)}, \tag{10}$$

where $\mathcal{N}_u^{I_j}$ is the set of all latent users who have interactions on item I_j, and $\mathcal{N}_I^{I_j}$ denotes the set of all neighboring items of I_j.

By adopting layer-wise message aggregation, the final representations of I_j and $u_{k,h}$ are denoted as follows:

$$\mathbf{E}_{I_j} = \sum_{l=0}^{L} \hat{\mathbf{C}}_{I_j}^{(l)}; \quad \mathbf{E}_{u_{k,h}} = \sum_{l=0}^{L} \hat{\mathbf{C}}_{u_{k,h}}^{(l)}, \tag{11}$$

where L is a hyper-parameter that controls the layer number of the graph convolutions on the primary capsule graphs.

Account-level Dynamic Routing. The account-level dynamic routing mechanism performs a routing selection to consider the associations between account and its latent users. The strength of the connections between user capsules and account capsules are qualified via a coupling coefficient b_p, which is initialized randomly. The dynamic routing is operated iteratively for θ times, where θ is a hyper-parameter. As a common practice (Zheng et al. 2022), we uniformly set θ as 3, maintaining a balance between performance and computational complexity. The dynamic routing at the j-th iteration is represented as Eqn. (12).

$$\tilde{\mathbf{C}}_{A_k}^{(j)} = \sum_{h}^{A_k} \text{squash}\left(\sum_{p} b_p^{(j-1)} \mathbf{E}_{u_{k,h}}\right), \tag{12}$$

where $\tilde{\mathbf{C}}_{A_k}^{(j)}$ denotes the account capsule for A_k, squash$(\cdot)$ is the squash function which compresses the routed information, ensuring efficient information transmission.

In addition, the coupling coefficient b_p at the j-th iteration is updated by calculating the affinity between user capsules and the account capsule:

$$b_p^{(j)} = b_p^{(j-1)} + \mathbf{W}_d \sum_{h}^{A_k} \left(\mathbf{E}_{u_{k,h}} \odot \tilde{\mathbf{C}}_{A_k}^{(j)}\right), \tag{13}$$

where $\mathbf{W}_d$ is a learnable weighting matrix, $\odot$ denotes the element-wise product.

Hence, the final representations of accounts and sequences are denoted as Eqn. (14).

$$\mathbf{E}_A = \sum_{A_k \in \mathcal{A}} \tilde{\mathbf{C}}_{A_k}^{(\theta)}; \quad \mathbf{E}_S = \sum_{S_k \in \mathcal{S}} \sum_{I_j \in S_k} \mathbf{E}_{I_j}. \tag{14}$$

With the help of the Graph Capsule Convolutional Network (GC^{2}N), the account representation has gained the ability to finely distinguish the preferences of various potential users within shared accounts. However, the diverse preferences within sequence representations remain largely unexplored. Hence, we further devise a subspace alignment method.

Subspace Alignment (SA)

Subspace alignment is an efficient component that clusters the hybrid preferences of multiple latent users and then aligns them to the sequence representations $\mathbf{E}_S$. Instead of using traditional self-representation matrices (Xie et al. 2022b; Zhang et al. 2018), SA exploits low-rank subspace bases to cluster diverse preferences for various latent users. Moreover, it also refines the sequence representations by attaching the subspace affinities, and then aligns them with the original sequence representations via a contrastive learning strategy. This strategy provides additional self-supervised signals to distinguish the preferences of latent users within hybrid sequences.

Subspace Affinity Calculation. Taking the sequence $S_k \in \mathcal{S}$ as an example, $\mathbf{E}_{S_k} \in \mathbb{R}^{n \times d_2}$ denotes its representations. The initialization of subspace bases $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_j, \ldots, \mathbf{d}_\alpha\} \in \mathbb{R}^{\alpha \times d_2}$ is given by the column space of the clusters generated by K-means on $\mathbf{E}_{S_k}$. Then, the subspace affinities are calculated as:

$$s_{ij} = \frac{\left\|\mathbf{e}_i^\top \mathbf{d}_j\right\|_F^2 + \lambda d_2}{\sum_j \left(\left\|\mathbf{e}_i^\top \mathbf{d}_j\right\|_F^2 + \lambda d_2\right)}, \tag{15}$$

where s_{ij} denotes the subspace affinity between i-th item $\mathbf{e}_i$ in S_k and j-th subspace base $\mathbf{d}_j$, λ is a parameter that controls the smoothness of the calculation (λ is uniformly set to 1e-4 according to the common practice reported in (Cai et al. 2022)).

As the affinity calculation does not rely on the self-expression framework, SA is able to achieve linear computational complexity (i.e., O($N \times d^2$)) and low memory consumption, making subspace clustering more efficient.

Contrastive Learning. To align the sequence embeddings with the clustered user preferences, SA first refines the $\mathbf{e}_i \in \mathbb{R}^{1 \times d_2}$ by applying the affinities:

$$\mathbf{z}_i = \mathbf{e}_i \cdot s_{ij}, \tag{16}$$

where $\mathbf{z}_i \in \mathbf{Z}_{S_k}$ denotes the refined embedding of i-th item in S_k and $\mathbf{Z}_{S_k} \in \mathbb{R}^{n \times d_2}$ represents the refined representation of S_k.

Then, SA adopts a contrastive learning paradigm that aligns the refined representations $\mathbf{Z}_{S_k}$ with the normalized sequence embeddings $\hat{\mathbf{E}}_{S_k}$:

$$\mathcal{L}_C^{S_k} = -\sum_{\mathbf{z}_i \in \mathbf{Z}_{S_k}}^{\hat{\mathbf{e}}_i \in \hat{\mathbf{E}}_{S_k}} \log\left(\frac{\exp(\hat{\mathbf{e}}_i \odot \mathbf{z}_i/\beta)}{\sum_{j \in \mathcal{I}} \exp(\hat{\mathbf{e}}_i \odot \mathbf{z}_j/\beta)}\right), \tag{17}$$

where $\mathcal{L}_C^{S_k}$ is the InfoNCE loss calculated between representations of S_k, $(\hat{\mathbf{e}}_i, \mathbf{z}_i)$ is a positive pair while $(\hat{\mathbf{e}}_i, \mathbf{z}_j)$ represents a negative pair, β denotes the temperature coefficient that controls the impact from negative pairs to positive pairs.

The contrastive loss for all the sequences is:

$$\mathcal{L}_C = \sum_{S_k \in \mathcal{S}} \mathcal{L}_C^{S_k}. \tag{18}$$

Such a strategy is able to distinguish various preferences of latent users. Therefore, it improves the capability of learning fine-grained sequence representations. The final sequence embeddings are obtained by summing the refined sequence embeddings and their normalized-forms:

$$\hat{\mathbf{E}}_S = \sum_{S_k \in \mathcal{S}} \text{Norm}\left(\hat{\mathbf{E}}_{S_k} + \mathbf{W}_s \mathbf{Z}_{S_k}\right), \tag{19}$$

where $\hat{\mathbf{E}}_S$ denotes the updated sequence embeddings, $\mathbf{W}_s$ is the weighting matrix.

Final Prediction

The final prediction generated by LightGC^{2}N is denoted as:

$$P(I_{t+1}|\mathcal{S}, \mathcal{A}) = \text{softmax}\left(\mathbf{W}_f \cdot [\hat{\mathbf{E}}_S, \mathbf{E}_A]^\top + \mathbf{b}_f\right), \tag{20}$$

where $\mathbf{W}_f$ is the transformation matrix that maps the predictions to the dimension of candidate items, and $\mathbf{b}_f$ is the bias term that adjusts the threshold of the activation function.

The cross-entropy loss function is adopted to optimize the learnable parameters in LightGC^{2}N, which is denoted as:

$$\mathcal{L}_S = -\frac{1}{|\mathcal{S}|} \sum_{I_{t+1} \in \mathcal{I}} \log P(I_{t+1}|\mathcal{S}, \mathcal{A}). \tag{21}$$

Then, the overall loss function is denoted as:

$$\mathcal{L} = \mathcal{L}_S + \gamma \mathcal{L}_C, \tag{22}$$

where γ is a hyper-parameter that controls the participation of self-supervised signals.

Experiments

In this section, we first introduce the experimental settings, and then analyze the performance of LightGC^{2}N by answering the following **R**esearch **Q**uestions.

- **RQ1:** How does the performance of LightGC^{2}N in terms of training efficiency and parameter scale?
- **RQ2:** How does LightGC^{2}N perform on the SSR compared with other state-of-the-art methods?
- **RQ3:** How do the key components of LightGC^{2}N contribute to the recommendation performance?
- **RQ4:** How do the hyper-parameters affect the performance of LightGC^{2}N?

Experimental Settings

Datasets. We evaluate LightGC^{2}N on four real-world datasets released by (Ma et al. 2019), including Hvideo-E (HV-E), Hvideo-V (HV-V), Hamazon-M (HA-M), and Hamazon-B (HA-B). HV-E and HV-V are two smart TV datasets comprising viewing logs from different TV channels. HV-E encompasses logs of educational videos and instructional content in areas such as sports nutrition and medicine, whereas HV-V includes logs of television series and films. HA-M and HA-B are derived from two Amazon domains, featuring movie viewing (HA-M) and book reading (HA-B). For the evaluation, we randomly assigned 80% of the sequences to the training set, and the remaining 20% to the testing set. Note that, the most recently observed item in each sequence per dataset is designated as the ground truth item.

Dataset	Metric	NCF	LightGCN	HRNN	NAIS	TGSRec	π-Net	PSJNet	DA-GCN	TiDA-GCN	**LightGC^{2}N**
HV-E	RC@5	11.25	20.70	22.55	19.80	19.91	25.13	24.80	51.35	54.11	**61.35**
	RC@20	20.12	39.92	47.98	40.17	41.80	47.08	46.68	66.93	68.98	**72.73**
	MRR@5	5.77	11.55	13.76	11.45	13.95	15.36	15.37	35.63	38.66	**46.24**
	MRR@20	7.85	13.56	16.14	13.24	15.73	17.52	17.56	37.27	40.23	**47.35**
HV-V	RC@5	27.21	58.45	68.00	59.30	58.91	67.00	66.86	75.39	76.37	**79.29**
	RC@20	22.63	63.55	73.24	67.41	67.22	74.17	74.14	82.37	83.58	**84.70**
	MRR@5	22.99	54.00	60.58	51.52	50.32	60.37	61.89	59.78	63.58	**66.27**
	MRR@20	24.71	56.72	63.31	54.47	53.40	61.74	62.63	60.55	65.37	**66.87**
HA-M	RC@5	7.82	15.54	16.96	14.03	14.59	18.54	16.25	22.93	23.55	**46.02**
	RC@20	10.34	18.20	20.81	16.02	18.44	21.87	18.14	23.90	24.33	**48.46**
	MRR@5	2.72	12.88	13.75	10.55	11.91	16.24	11.25	20.09	20.91	**41.30**
	MRR@20	3.11	13.12	14.14	12.57	14.00	16.56	13.58	20.19	21.23	**41.64**
HA-B	RC@5	8.31	21.14	20.92	14.51	14.57	22.44	16.67	23.93	24.69	**47.36**
	RC@20	11.22	22.88	23.64	19.82	19.93	23.75	19.30	24.25	24.82	**48.55**
	MRR@5	7.92	15.58	17.04	13.29	15.74	20.38	15.52	21.35	21.88	**44.40**
	MRR@20	9.88	17.30	17.35	15.99	16.63	20.58	17.30	21.39	22.21	**44.52**

Table 1: Experimental results (%) for different methods on four real-world datasets. The best results are indicated in bold, while underlined values indicate the sub-optimal results.

Evaluation Metrics. For model evaluations, we adopt two common evaluation metrics (Guo et al. 2021) to assess the model performance, i.e., top-N Recall (Recall@N) and top-N Mean Reciprocal Rank (MRR@N), where $N = \{5, 20\}$.

Implementation Details. We implemented LightGC^{2}N with TensorFlow and accelerated the model training using an Intel® Xeon® Silver 4210 CPU (2.20GHz) and NVIDIA® RTX 3090 (24G) GPU. The operating system is Ubuntu 22.04, the system memory is 126G, and the coding platform is PyCharm. The learnable parameters are initialized via Xavier (Glorot and Bengio 2010), the loss function is optimized by Adam (Kingma and Ba 2015) optimizer. For the training settings, we set the batch-size as 256, the learning rate as 0.005, the dropout rate as 0.1, and the training epochs as 200. We uniformly set the embedding-size as 16 for LightGC^{2}N and other baseline methods to ensure the fairness of experiments. For other hyper-parameters of baselines, we adopt optimal hyper-parameter settings reported in their paper and then fine-tuned them on each dataset.

Baselines. To validate the performance of LightGC^{2}N on SSR, we compared it with the following baselines: 1) Traditional recommendations: NCF (He et al. 2017), and LightGCN (He et al. 2020). 2) Sequential recommendations: HRNN (Quadrana et al. 2017). NAIS (He et al. 2018), and TGSRec (Fan et al. 2021). 3) Shared-account sequential recommendations: π-Net (Ma et al. 2019), PSJNet (Sun et al. 2023), DA-GCN (Guo et al. 2021), and TiDA-GCN (Guo et al. 2024).

Parameter Scale and Training Efficiency (RQ1)

In this section, we initially vary the proportion of input data from 0.2 to 1.0 on the HV-E and HV-V datasets to assess the LightGC^{2}N's training time consumption. Subsequently, we compare its parameter scale with other competitive methods, i.e., PSJNet, TiDA-GCN, π-net and DA-GCN. The observations are as follows: 1) Figure 3 (a) and (b) reveal that LightGC^{2}N exhibits reduced training time compared to other baselines, signifying enhanced training efficiency and scalability for large-scale datasets. 2) As depicted in Figure 3 (c) and (d), LightGC^{2}N requires notably fewer parameters than other methods, providing a positive answer to RQ1.

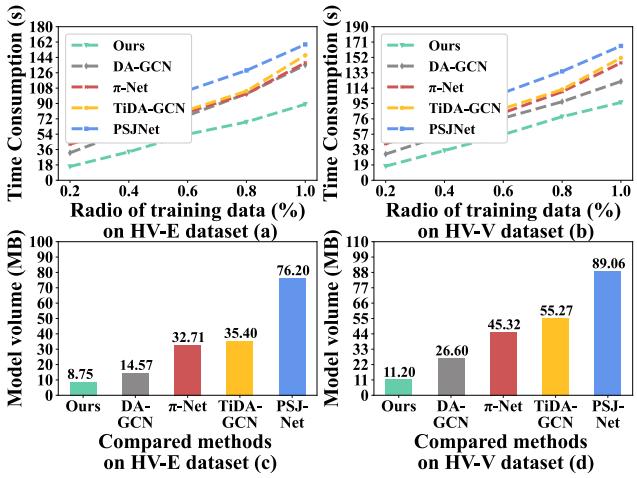

Figure 3: Comparison of time consumption and parameter scale between LightGC^{2}N and competitive SSR methods.

Overall Performance (RQ2)

Table 1 shows the experimental results of LightGC^{2}N compared with other state-of-the-art methods on four datasets. The observations are summarized as follows: 1) The sequential recommendation methods (i.e., HRNN, NAIS, and TGSRec) perform better than traditional methods (i.e., NCF and LightGCN). This observation indicates the significance of modeling users' sequential preferences. 2) The SSR so-

Dataset	HV-E				HV-V			
Metric	Recall		MRR		Recall		MRR	
	@5	@20	@5	@20	@5	@20	@5	@20
Light$_{w/o\text{LA}}$	58.33	67.58	39.71	40.24	76.01	81.61	63.17	64.04
Light$_{w/o\text{DR}}$	59.86	68.66	40.52	41.44	77.33	82.15	64.22	64.75
Light$_{w/o\text{C}}$	53.46	66.88	35.42	36.91	73.12	78.88	61.72	63.92
Light$_{w/o\text{CL}}$	59.96	71.12	43.51	44.68	77.02	83.15	64.85	65.10
Light$_{w/o\text{S}}$	57.96	70.12	40.51	41.68	75.12	82.16	62.88	64.05
Light$_{w/o\text{A}}$	22.90	43.42	16.26	22.97	60.58	65.30	54.94	57.82
LightGC^{2}N	**61.35**	**72.73**	**46.24**	**47.35**	**79.29**	**84.70**	**66.27**	**66.87**

Table 2: The experimental results (%) of ablation studies on two real-world datasets.

lutions (i.e., π-Net, PSJNet, DA-GCN, TiDA-GCN, and LightGC^{2}N) typically outperform the other traditional and sequential recommendation methods, demonstrating the significance of addressing the shared-account issues in real-world sequential recommendation scenarios. 3) LightGC^{2}N outperforms other state-of-the-art SSR methods (i.e., π-Net, PSJNet, DA-GCN, and TiDA-GCN). This observation indicates the significance of capturing the fine-grained differences among latent users for SSR. 4) LightGC^{2}N exceeds other graph-based SSR methods (i.e., DA-GCN and TiDA-GCN), demonstrating the superiority of our proposed graph capsule convolutional networks in modeling complicated associations for SSR. 5) LightGC^{2}N achieves the best performance on all datasets, demonstrating the superiority of our proposed graph capsule convolutional network and subspace alignment method for the SSR scenarios.

Ablation Study (RQ3)

We perform ablation studies on HV-E and HV-V to assess the impact of LightGC^{2}N's components. Table 2 presents variants: Lightw/oC (replaces GC^{2}N with a traditional network), Lightw/oS (disables Subspace Alignment), Lightw/oLA (uses Conv1D instead of Linear attention), Lightw/oDR (omits dynamic routing), Lightw/oCL (excludes contrastive learning), and Lightw/oA (removes all components).

The observations of Table 2 are summarized as follows: 1) LightGC^{2}N surpasses Lightw/oC and Lightw/oA, highlighting GC^{2}N's ability to discern user preferences and enhance SSR performance. 2) Superior performance over Lightw/oLA underscores Linear attention's role in capturing item correlations. 3) Outperforming Lightw/oDR indicates the effectiveness of account-level dynamic routing in integrating multi-user preferences. 4) Better results than Lightw/oS affirm the subspace alignment method's contribution to sequence representation. 5) LightGC^{2}N's advantage over Lightw/oCL demonstrates the benefits of contrastive learning in aligning sequence embeddings.

Hyper-parameters Analysis (RQ4)

The hyper-parameter α controls the number of latent users within each shared account. Figures 4 (a) and (b) show

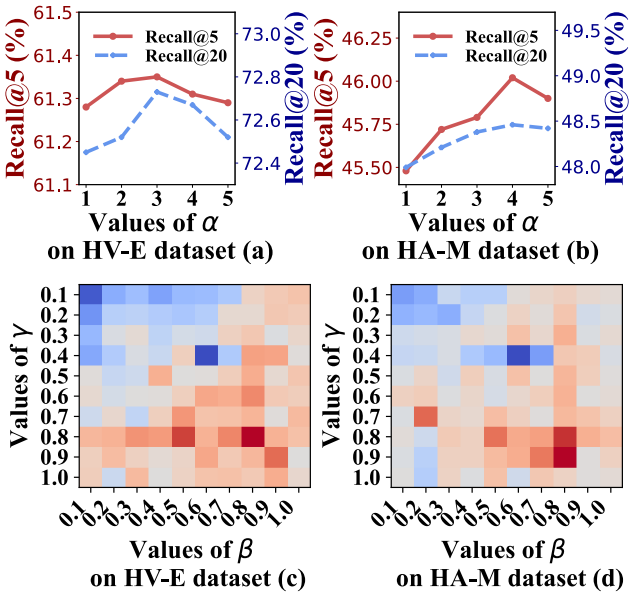

Figure 4: Impact of hyper-parameters α, β and γ on HV-E and HA-M.

the performance of LightGC^{2}N with different α values in $\{1, 2, 3, 4, 5\}$ on two different datasets (HV-E and HA-M). The experimental results indicate that LightGC^{2}N requires different parameter settings to achieve optimal performance on different datasets, which is consistent with the real-world setting (i.e., the number of latent users sharing an account varies in different scenarios). β and γ are two significant hyper-parameters that respectively control the temperature and the participation of the contrastive learning. As shown in Figures 4 (c) and (d), LightGC^{2}N reach the best performance when they are set to 0.8 or 0.9. This observation further validates the effectiveness of our contrastive learning strategy. Additionally, it indicates that the self-supervised signals contribute to sequence-level representation learning only when these hyper-parameters are appropriately valued.

Conclusion and Future Work

In this work, we introduce a Lightweight Graph Capsule Convolutional Network (LightGC^{2}N) with subspace alignment to tackle the problems in Shared-account Sequential Recommendation (SSR). By effectively capturing the fine-grained preferences of latent users, LightGC^{2}N achieves the best performance on various datasets. The lightweight design of this work makes it suitable for deployment on resource-constrained devices while maintaining high recommendation accuracy. Experimental results on four SSR datasets demonstrates the effectiveness and efficiency of LightGC^{2}N, paving the way for its practical application in real-world recommendation systems.

A promising future research direction is to enable LightGC^{2}N to automatically identify the number of latent users per shared account, obviating manual hyper-parameter tuning and improving model adaptability.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 62472263, 62072288), the Taishan Scholar Program of Shandong Province, Shandong Youth Innovation Team, the Natural Science Foundation of Shandong Province (Grant No. ZR2024MF034, ZR2022MF268).

References

Cai, C.; He, R.; and McAuley, J. J. 2017. SPMC: Socially-Aware Personalized Markov Chains for Sparse Sequential Recommendation. In *IJCAI*, 1476–1482. Melbourne, Australia: ijcai.org.

Cai, J.; Fan, J.; Guo, W.; Wang, S.; Zhang, Y.; and Zhang, Z. 2022. Efficient Deep Embedded Subspace Clustering. In *CVPR*, 21–30. New Orleans, LA, USA: IEEE.

Chen, J.; Zou, G.; Zhou, P.; Wu, Y.; Chen, Z.; Su, H.; Wang, H.; and Gong, Z. 2024. Sparse Enhanced Network: An Adversarial Generation Method for Robust Augmentation in Sequential Recommendation. In *AAAI*, 8283–8291. Vancouver, Canada: AAAI Press.

Fan, Z.; Liu, Z.; Zhang, J.; Xiong, Y.; Zheng, L.; and Yu, P. S. 2021. Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *CIKM*, 433–442. Queensland, Australia: ACM.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 249–256. Sardinia, Italy: JMLR.org.

Guo, L.; Tang, L.; Chen, T.; Zhu, L.; Nguyen, Q. V. H.; and Yin, H. 2021. DA-GCN: A Domain-aware Attentive Graph Convolution Network for Shared-account Cross-domain Sequential Recommendation. In *IJCAI*, 2483–2489. Montreal, Canada: ijcai.org.

Guo, L.; Zhang, J.; Chen, T.; Wang, X.; and Yin, H. 2023. Reinforcement Learning-Enhanced Shared-Account Cross-Domain Sequential Recommendation. *TKDE*, 35(7): 7397–7411.

Guo, L.; Zhang, J.; Tang, L.; Chen, T.; Zhu, L.; and Yin, H. 2024. Time Interval-Enhanced Graph Neural Network for Shared-Account Cross-Domain Sequential Recommendation. *TNNLs*, 35(3): 4002–4016.

He, R.; and McAuley, J. J. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *ICDM*, 191–200. Barcelona, Spain: IEEE Computer Society.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*, 639–648. Virtual Event, China: ACM.

He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.; and Chua, T. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *TKDE*, 30(12): 2354–2366.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T. 2017. Neural Collaborative Filtering. In *WWW*, 173–182. Perth, Australia: ACM.

Jiang, J.; Li, C.; Chen, Y.; and Wang, W. 2018. Identifying Users behind Shared Accounts in Online Streaming Services. In *SIGIR*, 65–74. Ann Arbor, MI, USA: ACM.

Kang, W.; and McAuley, J. J. 2018. Self-Attentive Sequential Recommendation. In *ICDM*, 197–206. Singapore: IEEE Computer Society.

Katharopoulos, A.; Vyas, A.; Pappas, N.; and Fleuret, F. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *ICML*, 5156–5165. Virtual Event: PMLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, 1–15. San Diego, CA, US: openreview.org.

Li, L.; Lian, J.; Zhou, X.; and Xie, X. 2024. Ada-Retrieval: An Adaptive Multi-Round Retrieval Paradigm for Sequential Recommendations. In *AAAI*, 8670–8678. Vancouver, Canada: AAAI Press.

Ma, H.; Xie, R.; Meng, L.; Chen, X.; Zhang, X.; Lin, L.; and Kang, Z. 2024. Plug-In Diffusion Model for Sequential Recommendation. In *AAAI*, 8886–8894. Vancouver, Canada: AAAI Press.

Ma, M.; Ren, P.; Lin, Y.; Chen, Z.; Ma, J.; and de Rijke, M. 2019. π-Net: A Parallel Information-sharing Network for Shared-account Cross-domain Sequential Recommendations. In *SIGIR*, 685–694. Paris, France: ACM.

Quadrana, M.; Karatzoglou, A.; Hidasi, B.; and Cremonesi, P. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*, 130–137. Como, Italy: ACM.

Shin, Y.; Choi, J.; Wi, H.; and Park, N. 2024. An Attentive Inductive Bias for Sequential Recommendation beyond the Self-Attention. In *AAAI*, 8984–8992. Vancouver, Canada: AAAI Press.

Sun, W.; Ma, M.; Ren, P.; Lin, Y.; Chen, Z.; Ren, Z.; Ma, J.; and de Rijke, M. 2023. Parallel Split-Join Networks for Shared Account Cross-Domain Sequential Recommendations. *TKDE*, 35(4): 4106–4123.

Verstrepen, K.; and Goethals, B. 2015. Top-N Recommendation for Shared Accounts. In *RecSys*, 59–66. Vienna, Austria: ACM.

Wen, X.; Peng, Z.; Huang, S.; Wang, S.; and Yu, P. S. 2021. MISS: A Multi-user Identification Network for Shared-Account Session-Aware Recommendation. In *DASFAA*, 228–243. Taipei, Taiwan: Springer.

Wu, B.; He, X.; Zhang, Q.; Wang, M.; and Ye, Y. 2023. GCRec: Graph-Augmented Capsule Network for Next-Item Recommendation. *TNNLs*, 34(12): 10164–10177.

Xie, X.; Sun, F.; Liu, Z.; Wu, S.; Gao, J.; Zhang, J.; Ding, B.; and Cui, B. 2022a. Contrastive Learning for Sequential Recommendation. In *ICDE*, 1259–1273. Kuala Lumpur, Malaysia: IEEE.

Xie, Y.; Li, W.; Sun, Y.; Bertino, E.; and Gong, B. 2022b. Subspace Embedding Based New Paper Recommendation. In *ICDE*, 1767–1780. Kuala Lumpur, Malaysia: IEEE.

Yue, G.; Xiao, R.; Zhao, Z.; and Li, C. 2023. AF-GCN: Attribute-Fusing Graph Convolution Network for Recommendation. *TBD*, 9(2): 597–607.

Zhang, Q.; Lu, J.; Wu, D.; and Zhang, G. 2018. Cross-domain Recommendation with Consistent Knowledge Transfer by Subspace Alignment. In *WISE*, 67–82. Dubai, United Arab Emirates: Springer.

Zheng, X.; Liang, X.; Wu, B.; Guo, Y.; and Zhang, X. 2022. Graph Capsule Network with a Dual Adaptive Mechanism. In *SIGIR*, 1859–1864. Madrid, Spain: ACM.