GS^2-RS: A Generative Approach for Alleviating Cold Start and Filter Bubbles in Recommender Systems

Yuanbo Xu, En Wang, Yongjian Yang, and Hui Xiong, *Fellow, IEEE*

***Abstract*—Recommender Systems (RSs) typically face the cold-start problem and the filter-bubble problem when users suffer the familiar, repeated, and even predictable recommendations, making them bored and unsatisfied. The key to solving these issues is learning users' fine-grained preferences and recommending appealing and unexplored items deviating from users' historical items. However, existing models consider cold-start or filter bubble problems separately and ignore that they can reinforce mutually and damage the models' performance accuracy. To this end, we devise a novel serendipity-oriented recommender system (Generative Self-constrained Serendipitous Recommender System, GS^2-RS) that generates users' fine-grained preferences to enhance the recommendation performance. Specifically, GS^2-RS extracts users' interest and satisfaction preferences and generates virtual but convincible neighbors' preferences from themselves with a twin Conditional Generative Adversarial Nets (not from real neighbors). Then we introduce the serendipity item, which is low-interest but high-satisfaction among candidate items. We use the serendipity item to improve the diversity of recommended items, which relieves the filter-bubble problem. Along with this line, a gated mechanism is applied to their fine-grained preferences (interests, satisfactions) to obtain their serendipity items. Finally, these serendipity items are inversely injected into the original user-item rating matrix and build a relatively dense matrix as the input for backbone RS models. Note that GS^2-RS tackles cold-start and filter-bubble problems in a unified framework without any additional side information and enriches the interpretability of recommendation models. We comprehensively validate GS^2-RS for solving cold-start and filter bubble problems on four real-world benchmark datasets. Extensive experiments illustrate GS^2-RS's superiority in accuracy, serendipity, and interpretability over state-of-the-art models. Also, we can plug our model into existing recommender systems as a preprocessing procedure to enhance their performance.**

***Index Terms*—Generative models, cold start, filter bubble, recommendation.**

Manuscript received 11 November 2022; revised 2 June 2023; accepted 18 June 2023. Date of publication 28 June 2023; date of current version 11 January 2024. This work was supported in part by the National Natural Science Foundations of China under Grants 61772230, 61976102, U19A2065, and 61972450, in part by the Natural Science Foundation of China for Young Scholars under Grants 61702215 and 62002132, in part by Jilin Education Science Foundation under Grant JJKH20221010KJ, and in part by Jilin Science and Technology Research Project under Grant 20230101067JC. Recommended for acceptance by T. Weninger. *(Corresponding author: En Wang.)*

Yuanbo Xu, En Wang, and Yongjian Yang are with the Mobile Intelligent Computing (MIC) Lab, Key Laboratory of Symbolic Computation and Knowledge Engineering for the Ministry of Education, Department of Computer Science and Technology, Jilin University, Changchun 130012, China (e-mail: yuanbox@jlu.edu.cn; wangen0310@126.com; yyj@jlu.edu.cn).

Hui Xiong is with the Thrust of Artificial Intelligence, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 510230, China, and also with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China (e-mail: xionghui@ust.hk).

Digital Object Identifier 10.1109/TKDE.2023.3290140

I. Introduction

As an important tool to filter the enormous amount of information, the recommender system selects relevant candidate items for object users while simultaneously extracting their personalized preferences from their historical shopping logs. Collaborative filtering (CF) [1] and matrix factorization (MF) [2] have been the most popular algorithms for recommender systems. However, as reported in the recent research literature, conventional CF and MF models often suffer from cold-start situations [3], filter bubbles [4], and overfitting issues [5], which results in inaccurate, irrelevant, and repeated recommendations. Moreover, most backbone recommender systems focus on enhancing the experimental accuracy, such as Hitting ratio, Recall, etc., with complicated networks, powerful computing ability, and various side information. These models' strong fitting ability may lead to an overlook on analyzing the cold-start and filter bubble problems from a data perspective. We should explore the insights between users and items to solve real-world issues (cold-start, filter bubbles, etc.) rather than directly employ the most powerful models or frameworks to fit the feedback. Along with this line, we try to simultaneously analyze and solve two crucial issues: cold-start and filter-bubble problems, from a data-driven perspective, with a simple and explainable framework.

Cold-Start (CS) problem and *Filter-Bubble* (FB) problem are the vital issues that limit the performance of recommender systems, as the exhibition in Fig. 1. CS problem is a common but great challenge because the feedback interactions (clicks, browses, purchases, ratings, etc. In this paper, we focus on ratings.) only account for a minimal fraction between millions of items and users. We define the users and items without enough interactions as the Cold-start users and Cold-start items. The core issue of CS problem is that the high sparsity of user-item interactions offers limited information about Cold-start users and items. Obviously, it is difficult for RS models to predict these users' preferences without enough historical ratings, thus leading to inaccurate recommendations. As a compromise, existing models usually recommend popular items [6] to Cold-start users. Nevertheless, recommending similar, repeated popular items without personalization magnifies another critical issue- the problem of filter bubbles (FB). Different from echo chambers [7] or information cocoons [8], the Filter bubble is caused by the recommender system or recommend methodology that the users would be represented by their historical interactions. Thus, the recommendations would be more and more centralized among their historical interactions, forming the filter bubbles as reported

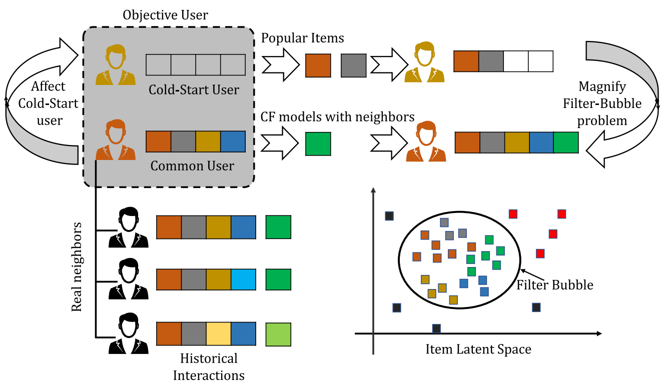


Fig. 1. An example to explain CS and FB problems in real-world scenarios (better viewed in color mode). We recommend Top-1 item for the common user and Top-popular-2 items for the Cold-start user. Note that recommendations for the Cold-start user add the popular items' interaction counts (making popular items more popular) and trend to homogenize the users' recommendation, which magnifies the filter bubble situation. At the same time, the similar items in filter bubbles affect the Cold-start user's chances of finding his/her taste (Next, the green item is possible to be recommended to him/her). Both CS and FB problems reinforce each other.

in Fig. 1. Naturally, these homogeneous items in filter bubbles have more chances to be exposed to users than the other items should, which is a typical scenario of the *Matthew Effect*. Worse, by recommending the popular items to Cold-start users as a normal solution, these items become too impactful to magnify the filter-bubble situation, dominating the recommendation results. Correspondingly, the increasing interactions with popular items affect the recommendation to Cold-start users and vice versa. In this way, the CS and FB problems usually appear in pairs and reinforce mutually, which seriously impacts the recommendation quality.

There are some existing solutions for solving Cold-start or Filter-bubble problems, respectively. For the cold-start problem, some studies employed novel data frameworks to explore the latent knowledge from side information (attributes, features, etc.) and develop the existing recommendation models. These powerful models, such as multi-layer perceptron [9], recurrent neural networks [10] or the latest graph convolutional network [11], were proposed to find the latent neighbors or the similar latent representations of cold-start users/items. However, side information is difficult to obtain due to privacy restrictions, limiting these models' application prospects. In this context, *data completion* has been utilized to tackle the CS problem without any side information [3], [6], [12] by inferring the data distribution from existing interactions. Data completion provides a direct, simple, and easy-to-deploy method for recommendations, while it is usually too coarse to give a reasonable recommendation (for example, they cannot extract users' preferences on items, lack of interpretability). For the filter-bubble problem, most researchers so far have focused on exploiting auxiliary information such as users' attributes [13], user's social relations [14], and item's description text [3] or reviews [15]. Along this line, serendipity-oriented recommender systems are proposed to make unexpected but valuable items to users [6], [16]. These researchers utilize a designed reward function to train models for adding diversity and serendipity to the recommendation list. Note that their models are effective and useful only when auxiliary information is available. But over-utilizing the auxiliary information may cause the privacy disclosure problem [17]. Besides, these methods only focus on solving either CS problem or FB problem without considering the inner relations between them. In this work, our research focuses on *DATA*: one user-item rating matrix; *Task*: a Top-k recommendation task without requiring any auxiliary information or changing the existing recommendation models. *Objective*: solving both CS and FB problems in a unified framework.

In this paper, we propose a novel recommender system framework that can *tackle the cold-start problem and the filter-bubble problem at the same time*. Different from existing neural network-based models or data completion models, our idea is to generate *virtual but convincible users' preferences from themselves*, drop out the impossible items from the candidate item set, thus improving the recommendation performance. Specifically, **GS^2-RS**, which stands for **G**enerative **S**elf-constrained **S**erendipitous **R**ecommendation **S**ystem is an end-to-end framework for tackling CS and FB problems. This model first analyzes the user-item rating matrix to form users' historical interest and satisfaction preferences granularly. Next, we train different Conditional Generative Adversarial Nets (CGANs) to generate users' virtual preferences (interest and satisfaction) from themselves. Note that we generate the preferences, not ratings. Then, we devise a gate mechanism to combine users' preferences to form their self-serendipity preferences and mark the impossible items in the user-item rating matrix to build an enhanced rating matrix. In this way, we obtain the serendipity for each item in an explainable way, and we could tune the gate threshold to form a personalized model. Finally, GS^2-RS outputs this enhanced rating matrix to existing state-of-the-art recommendation models as their new inputs or directly gives the recommendation list with simple CF models. To the best of our knowledge, our work is the first attempt to employ CGANs to tackle both CS and FB problems in a unified framework for recommender systems. Notably, the contributions are summarized as follows:

- We analyze two crucial problems: Cold-start and Filter-bubble problems, which are always considered respectively, but in fact, they reinforce mutually. So we propose a novel unified recommender system framework (GS^2-RS), which can tackle both CS and FB problems without any auxiliary information.
- For tackling the CS problem, GS^2-RS utilizes CGANs to generate users' virtual preferences from themselves for reliability and relieves data sparsity with a delicate matrix injection method to filter the candidate items. For tackling the FB problem, GS^2-RS extracts fine-grained preferences (interest, satisfaction, and serendipity) for the recommendation trade-off, adding the recommendation's diversity and interpretability with a stable accuracy.
- As an end-to-end framework, GS^2-RS can be treated as a preprocessing module for the state-of-the-art recommendation models. Moreover, we provide insight into mining the knowledge from user-item interactions by combining a

generative form with users' preferences. We also propose some optimization strategies for GS^2-RS.

- We conduct extensive empirical studies on several public datasets and find that GS^2-RS can achieve a superior recommendation performance on accuracy and serendipity metrics. Also, it provides explainable recommendations for each user.

II. Preliminary

Given a recommender system, which contains m users and n items, the user-item rating matrix R is a $m \times n$ low-rank sparse matrix, where its entity r_{ij} stands for the rating that user i marked for item j, ranging from [*1,2,3,4,5*]. Note that in real-world scenarios, R is usually extremely sparse (more than 90% data is unknown to learning models), meaning there are many "?" in R.

In this work, we focus on *original rating-based RS*, which means that for achieving a Top-*k* recommendation task, the input of RS models is only the original user-item rating matrix R. However, the sparsity of R usually leads to the cold-start (CS) problem and the filter-bubble (FB) problem [3], as the exhibition in Fig. 1. Specifically, Cold-start and Filter-bubble problems on R are defined as follows:

Definition 1. Cold-start problem: For a user u or an item i in user-item rating R, cold-start problem means that there are only c historical interactions available in recommender system, where $0 \leq c \ll \min(m,n)$. Lack of historical interactions (the extreme sparsity of R) may confuse RS models to infer inaccurate preferences of users on candidate items (user-based CS problem) or ignore the unpopular items (item-based CS problem), which damage the models' performance. Thus the key to CS problem is to *provide more interactions and relieve the sparsity issue in user-item rating matrix.*

Definition 2. Filter-bubble problem: Filter-bubble problem usually occurs on the system level among a group of users. If a user expresses his preferences on some items with strong correlations (items with the same topic, same category, etc.), recommender systems would make homogeneous recommendations for achieving a better prediction accuracy expectation. In the worst situation, for the users in a filter bubble, recommender systems can not provide any item with additional information entropy out of the filter bubble. For a Top-*k* recommendation $Rec = \{i_i, i_2, \ldots, i_k\}$ to a user u in a filter bubble FB with f items $\{fb_1, fb_2, \ldots, fb_f\}$, the information entropy gain is defined as Formula (1)

$$\text{I-gain}(Rec) = -\sum_{j=1}^{k} p(i_j)\log(p(i_j)), \tag{1}$$

where $p(i_j)$ could be calculated as the standard metric ($similarity$, $\frac{1}{diversity}$, or $\frac{1}{Euc-distance}$, etc.) between i_j and $FB = \{fb_1, fb_2, \ldots, fb_f\}$. The FB problem is defined as: A recommender system can not provide valuable recommendation Rec to users in a filter bubble FB, which leads to I-gain(Rec)=0. It is an intuitive solution for the FB problem that we can minimize the standard metric $p(i_j)$. But there are two limitations to this solution: 1) the item distribution in the filter bubble is unknown to the recommender systems. The scale f and the composition of the filter bubble usually vary with context and scenarios. And 2) the computing complexity is increasing exponentially compared with other recommender systems ($\mathcal{O}(mk^f)$ to $\mathcal{O}(mk)$). Along with this line, we loose the restriction of the standard metric $p(i_j)$: we define $\hat{p}(i_j)$ as the metric between i_j and $Rec = \{i_i, i_2, \ldots, i_k\}$, replacing $p(i_j)$ in Formula (1). The reason is that if the objective user u was in the filter bubble FB, the model would make a homogeneous recommendation with similar distribution as FB. By minimizing $\hat{p}(i_j)$, we could add the KL divergence between Rec and FB, thus relieving the FB problem. Thus, we transfer the FB problem to *adding user-level diversity in recommendation results.*

For solving CS and FB problems jointly, we notice that *user-level preference* is the core factor: With accurate user preference, we can filter the candidate items and drop out the most impossible items, which relieves the data sparsity issue; Meanwhile, we could treat the preferences as $\hat{p}(i_j)$ to add the diversity of recommendations.

Then we define user-level *serendipity preference* with two fine-grained preferences: *interest and satisfaction preferences:*

Definition 3. Interest and Satisfaction: We define both definitions as [16]. *Interest*: for a user u and an item i in a candidate itemset $i \in \mathbf{I}_u^{\text{can}}$, if $r_{ui} \neq ?$, which means that u has the interest to buy i, we define user' interest preference $r_{ui}^{\text{in}} = 1$, $r_{ui}^{\text{in}} \in \mathbf{R}^{\text{in}}$, $i \in \mathbf{I}_u^{\text{in}}$. *Satisfaction*: for a user u and an item $i \in \mathbf{I}_u^{\text{in}}$, if $r_{ui} \geq \alpha_{ui}$, we define user's satisfaction preference $r_{ui}^{\text{sa}} = 1$, $i \in \mathbf{I}_u^{\text{sa}}$; else $r_{ui}^{\text{sa}} = 0$, $r_{ui}^{\text{sa}} \in \mathbf{R}^{\text{sa}}$. Note that the threshold α_{ui} can be defined as either u or i's average rating or based on some other context. In our proposed model, we extract users' interest and satisfaction preferences from historical ratings in $\mathbf{R}$.

At last, we introduce the user-level serendipity preference, which is a combination of Interest and Satisfaction:

Definition 4. Serendipity: the items with high relevance but low shopping purpose [16], which means low Interest but high Satisfaction. In most recommender systems, the algorithm ignores the fact that for each specific user, the serendipity item may create more value than other popular items. Intuitively, serendipity item is always beyond the user's filter bubble range. However, note that there is no ground truth for serendipity items in real-world datasets. To achieve Serendipity, we predict Interest and Satisfaction for unrated items with historical Interest and Satisfaction. Thus, by adding serendipity items into Top-*k* results, the recommender system can achieve better diversity and satisfaction for solving the filter-bubble problem effectively. We aim to filter the candidate itemset with user-level interest, satisfaction, and serendipity preferences jointly, relieve the data sparsity issue for solving the CS problem, and propose an explainable model for solving the FB problem.

III. Framework of GS^2-RS

In this section, we introduce GS^2-RS as a unified framework for tackling CS and FB problems (Fig. 2). The input (original input $\mathbf{R}$) and output (enhanced matrix $\mathbf{R}^{\text{h}}$) of GS^2-RS are matrices with same dimensions. $\mathbf{R}$ is a $m \times n$ dimension

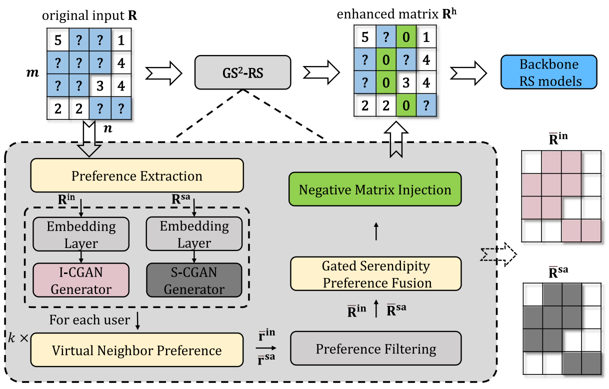


Fig. 2. Illustration of our proposed GS^2-RS framework.

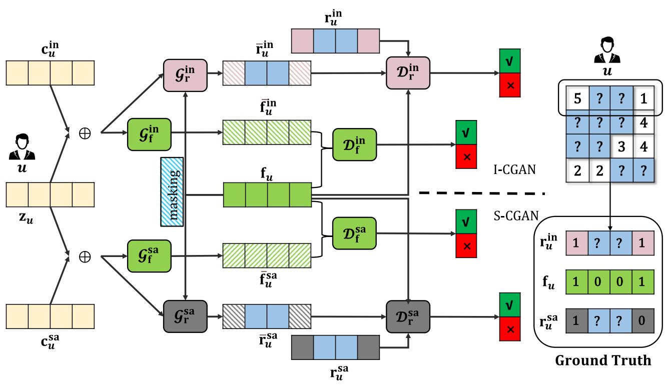


Fig. 3. Training process of Twin-CGAN in GS^2-RS.

user-item rating matrix. First, GS^2-RS utilizes a Preference Extraction module to build two matrices: interest matrix $\mathbf{R}^{\text{in}}$ and satisfaction matrix $\mathbf{R}^{\text{sa}}$, following their definitions (Definition 3). Then we input both matrices into two embedding layers to calculate the object user's embedding. We design a Twin-CGAN (I-CGAN and S-CGAN) to compute users' virtual preferences in a generative way. For each user u, we obtain k virtual neighbor preferences and use them to predict unrated items' Interest and Satisfaction preferences (the "?" in $\mathbf{R}$) to form $\overline{\mathbf{R}}^{\text{in}}$ and $\overline{\mathbf{R}}^{\text{sa}}$. Then a gated-serendipity preference fusion module is proposed to tackle both matrices. Finally, instead of picking up the potential items from candidate items, we utilize Negative Matrix Injection on $\overline{\mathbf{R}}^{\text{in}}$ and $\overline{\mathbf{R}}^{\text{sa}}$ to filter the candidate items and drop out the most impossible items to form enhanced matrix $\mathbf{R}^{\text{h}}$. Specifically, GS^2-RS could relieve the sparsity issue by adding 0 as historical interactions for tackling the CS problem, and the prediction of Interest, Satisfaction and Serendipity can be the ranking reason to add diversity in recommendation lists for tackling the FB problem. More details are given in the following section.

IV. Methodology of GS^2-RS

In this section, we introduce three key modules of GS^2-RS: A Twin-CGAN for predicting fine-grained preferences, a gated serendipity preference fusion to obtain serendipity preference, and a negative matrix injection to achieve the enhanced matrix. Also, we give an analysis of how GS^2-RS can enhance existing recommendation models.

A. Fine-Grained Preference Extraction and Prediction

We deduce fine-grained user-level preferences, including users' virtual interest and satisfaction preferences, based on Conditional Generative Adversarial Nets (CGAN) [18], a framework to train generative models with complicated, high-dimensional real-world data such as images. Specifically, CGAN is an extension to the original GAN [19]: it allows a generative model $\mathcal{G}$ to produce data according to a specific condition vector $\mathbf{c}$ by treating the desired condition vector as an additional input with the random noise input $\mathbf{z}$. Thus, CGAN's objective function is formulated as follows:

$$V(\mathcal{D},\mathcal{G}) = \mathbb{E}_{\mathbf{x}-p_{\text{data}}(\mathbf{x})}[\ln \mathcal{D}(\mathbf{x}|\mathbf{c})] - \mathbb{E}_{\mathbf{z}-p_{\mathbf{z}}(\mathbf{z})}[\ln \mathcal{D}(\mathcal{G}(\mathbf{z}|\mathbf{c}))], \tag{2}$$

where $\mathbf{x}$ is a ground truth data from the data distribution p_{data}, $\mathbf{z}$ is a noise input vector sampled from known prior $p_{\mathbf{z}}$. $\mathcal{G}(\mathbf{z})$ is a synthetic data from the generator distribution $p_{\mathbf{g}}$. $\mathbf{c}$ corresponds to a condition vector, such as a one-hot vector of a specific class label. With the optimal objective $\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D},\mathcal{G})$, ideally, the completely trained $\mathcal{G}$ is expected to generate realistic data, and the discriminative model $\mathcal{D}$ is expected to evaluate the possibility that the data came from the ground truth rather than from $\mathcal{G}$ equals 0.5.

In our model, we propose a Twin-CGAN framework, where we jointly train two CGANs for users' interest and satisfaction preferences $R^{\text{in}}, R^{\text{sa}}$. And we employ well-trained generators to generate virtual fine-grained virtual preferences in GS^2-RS. Fig. 3 explains the training process of Twin-CGAN. The motivations for designing the Twin-CGAN are 1) users' preferences should be distinct from other users, which is reasonable to generate their virtual preferences from themselves. 2) A generative model can introduce limited indeterminacy for forming users' preferences. Generative models aim at constructing a virtual data vector with a similar data distribution as the real one. Note that it is "similar", not "same", which means that there are some unexpected items that would be treated as users' candidate items. Intuitively, these unexpected but relative items are more suitable for our purpose of solving the filter-bubble problem by introducing more diversity into recommendations. Note that we apply the same CGAN framework on interest and satisfaction preferences, respectively. For the sake of simplicity, we take Interest-CGAN (the upper part of Fig. 3) as the example and briefly introduce how this framework can be properly deployed on satisfaction preferences. Formally, we build our Twin-CGAN as follows:

$$V(\mathcal{D}_{\text{r}}^{\text{in}}, \mathcal{G}_{\text{r}}^{\text{in}}, \mathcal{D}_{\text{f}}^{\text{in}}, \mathcal{G}_{\text{f}}^{\text{in}}) \simeq \frac{1}{|\mathbf{U}|}\Bigg(\sum_{u\in\mathbf{U}} (\ln \mathcal{D}_{\text{r}}^{\text{in}}(\mathbf{r}_u^{\text{in}}|\mathbf{c}_u^{\text{in}}) - \ln \mathcal{D}_{\text{r}}^{\text{in}}(\mathcal{G}_{\text{r}}^{\text{in}}(\mathbf{z}_u^{\text{in}}|\mathbf{c}_u^{\text{in}}) \bullet \mathrm{f}_u^{\text{in}}))\Bigg)$$

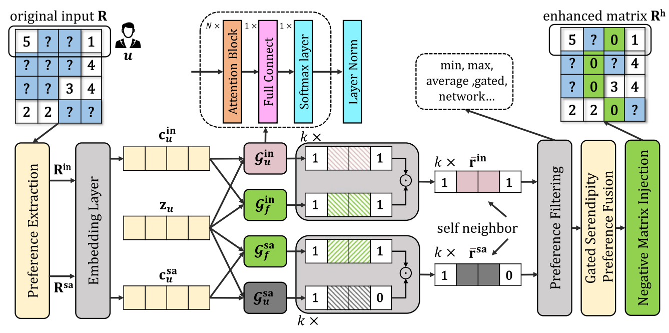


Fig. 4. GS^2-RS for enhancing recommender systems.

$$+\frac{1}{|\mathbf{U}|}\left(\sum_{u\in U}(\ln\mathcal{D}_{\mathrm{f}}^{\mathrm{in}}(\mathbf{f}_u^{\mathrm{in}}|\mathbf{c}_u^{\mathrm{in}}) - \ln\mathcal{D}_{\mathrm{f}}^{\mathrm{in}}(\mathcal{G}_{\mathrm{f}}^{\mathrm{in}}(\mathbf{z}_u^{\mathrm{in}}|\mathbf{c}_u^{\mathrm{in}})))\right), \tag{3}$$

where $\mathbf{c}_u^{\mathrm{in}}$ denotes u's interest condition vector, $\mathbf{z}_u^{\mathrm{in}}$ denotes u's latent embedding vector, $\mathbf{r}_u^{\mathrm{in}}$ denotes u's interest vector, $\mathbf{f}_u^{\mathrm{in}}$ denotes the interest indicator vector, $\bullet$ denotes a masking operation. Note that $\mathbf{f}_u^{\mathrm{in}}$ is in the same dimension as $\mathbf{r}_u^{\mathrm{in}}$, and each entity f_{ui}^{in} of $\mathbf{f}_u^{\mathrm{in}}$ is $1/0$ to indicate that there is/not an Interest value for item i.

To train the Twin-CGAN, we first obtain $\mathbf{c}_u^{\mathbf{in}}$ and $\mathbf{c}_u^{\mathbf{sa}}$, which are the output of the Embedding layer, respectively. Note that $\mathbf{c}_u^{\mathbf{in}}$ and $\mathbf{c}_u^{\mathbf{sa}}$ are fixed for each user u in a batch. Then we initialize $\mathbf{z}_u$ from a standard normal distribution $p_{\mathbf{z}_u}$

$$\begin{aligned}\mathbf{c}_u^{\mathbf{in}} &= \text{Embedding-Layer}(\mathbf{R}^{\mathbf{in}});\\ \mathbf{c}_u^{\mathbf{sa}} &= \text{Embedding-Layer}(\mathbf{R}^{\mathbf{sa}});\\ \mathbf{z}_u &\sim p_{\mathbf{z}_u}.\end{aligned} \tag{4}$$

Then we input $\mathbf{c}_u^{\mathbf{in}} \oplus \mathbf{z}_u$ into $\mathcal{G}_{\mathrm{r}}^{\mathrm{in}}$ and $\mathcal{G}_{\mathrm{f}}^{\mathrm{in}}$, respectively. $\oplus$ is a concatenation operation for vectors. In I-CGAN, $\mathcal{G}_{\mathrm{r}}^{\mathrm{in}}$ and $\mathcal{G}_{\mathrm{f}}^{\mathrm{in}}$ are employed to produce u's synthetic interest vector and interest indicator vector, denoted as $\overline{\mathbf{r}}_u^{\mathrm{in}}, \overline{\mathbf{f}}_u^{\mathrm{in}}$. Next, $\mathcal{D}_{\mathrm{r}}^{\mathrm{in}}$ and $\mathcal{D}_{\mathrm{f}}^{\mathrm{in}}$ are employed to distinguish real interest vector $\mathbf{r}_u^{\mathrm{in}}$, indicator vector $\mathbf{f}_u^{\mathrm{in}}$ from synthetic vectors $\overline{\mathbf{r}}_u^{\mathrm{in}}$, $\overline{\mathbf{f}}_u^{\mathrm{in}}$, respectively. For training I-CGAN, we extract ground truth $\mathbf{r}_u^{\mathbf{in}}, \mathbf{f}_u^{\mathbf{in}}$ from original user-item rating matrix $\mathbf{R}$. $\mathbf{f}_u^{\mathbf{in}}$ denotes the possibility of an item being exposed to the user u. Note that the generator and the discriminator have the same structure: N attention blocks, 1 full connect layer, and 1 softmax layer (Fig. 4). Specifically, there are two novel designs in this GAN-based framework: first, each $\mathcal{G}$'s outputs' values are restricted to the range of (0 to 1) for computing with a Layer Normalization; second, a masking operation is employed to avoid useless computations: $(\mathbf{z}_u^{\mathrm{in}}|\mathbf{c}_u^{\mathrm{in}}) \bullet \mathbf{f}_u^{\mathrm{in}}$. This element-wise dot operation forces the discriminator to focus on the values r_{ui}^{in} in the interest vector $\mathbf{r}_u^{\mathrm{in}}$, which contributes more to the computing results. Besides, the masking operation relieves the data sparsity issue. Above $\mathcal{D}, \mathcal{G}$ are co-trained where the stochastic gradient descent (SGD) with mini-batches and the back-propagation algorithm is employed for the optimization.

Similar to interest preferences, we build the same framework as S-CGAN, train it with ground truth $(\mathbf{r}_u^{\mathbf{sa}}, \mathbf{f}_u^{\mathbf{sa}})$ from $\mathbf{R}$. Satisfaction preferences can be modeled as follows:

$$\begin{aligned} &V(\mathcal{D}_{\mathrm{r}}^{\mathrm{sa}}, \mathcal{G}_{\mathrm{r}}^{\mathrm{sa}}, \mathcal{D}_{\mathrm{f}}^{\mathrm{sa}}, \mathcal{G}_{\mathrm{f}}^{\mathrm{sa}})\\ &\simeq \frac{1}{|\mathbf{U}|}\left(\sum_{u\in \mathbf{U}}(\ln\mathcal{D}_{\mathrm{r}}^{\mathrm{sa}}(\mathbf{r}_u^{\mathrm{sa}}|\mathbf{c}_u^{\mathrm{sa}}) - \ln\mathcal{D}_{\mathrm{r}}^{\mathrm{sa}}(\mathcal{G}_{\mathrm{r}}^{\mathrm{sa}}(\mathbf{z}_u^{\mathrm{sa}}|\mathbf{c}_u^{\mathrm{sa}})\bullet\mathbf{f}_u^{\mathrm{sa}}))\right)\\ &+\frac{1}{|\mathbf{U}|}\left(\sum_{u\in \mathbf{U}}(\ln\mathcal{D}_{\mathrm{f}}^{\mathrm{sa}}(\mathbf{f}_u^{\mathrm{sa}}|\mathbf{c}_u^{\mathrm{sa}}) - \ln\mathcal{D}_{\mathrm{f}}^{\mathrm{sa}}(\mathcal{G}_{\mathrm{f}}^{\mathrm{sa}}(\mathbf{z}_u^{\mathrm{sa}}|\mathbf{c}_u^{\mathrm{sa}})))\right). \end{aligned} \tag{5}$$

Note that when training the GAN, we set $\mathbf{f}_u^{\mathbf{in}} = \mathbf{f}_u^{\mathbf{sa}} = \mathbf{f}_u$. After the training of Twin-CGAN, we are ready to generate users' virtual preferences, including interest and satisfaction with four well-trained generators: $\mathcal{G}_{\mathrm{r}}^{\mathrm{in}}$, $\mathcal{G}_{\mathrm{f}}^{\mathrm{in}}$, $\mathcal{G}_{\mathrm{r}}^{\mathrm{sa}}$, $\mathcal{G}_{\mathrm{f}}^{\mathrm{sa}}$. With these virtual preferences, we aim to deduce users' self-serendipity preferences for an accurate recommendation.

B. Gated Serendipity Preference Fusion and Negative Matrix Injection

Fig. 4 exhibits the usage of GS^2-RS. With four well-trained generators $\mathcal{G}_{\mathrm{r}}^{\mathrm{in}}$, $\mathcal{G}_{\mathrm{f}}^{\mathrm{in}}$, $\mathcal{G}_{\mathrm{r}}^{\mathrm{sa}}$, $\mathcal{G}_{\mathrm{f}}^{\mathrm{sa}}$, we feed objective user u's condition vectors $\mathbf{c}_u^{\mathrm{in}}$, $\mathbf{c}_u^{\mathrm{sa}}$, and user latent vector $\mathbf{z}_u$, then we obtain the synthetic vectors $\overline{\mathbf{r}}_u^{\mathrm{in}}$, $\overline{\mathbf{f}}_u^{\mathrm{in}}$, $\overline{\mathbf{r}}_u^{\mathrm{sa}}$, $\overline{\mathbf{f}}_u^{\mathrm{sa}}$, which are formulated as follows:

$$\mathcal{G}_{u/\mathrm{f}}^{*}(\mathbf{z}_u^*, \mathbf{c}_u^*) = <\overline{\mathbf{r}}_u^*, \overline{\mathbf{f}}_u^* >, \tag{6}$$

where * denotes either $^{\mathrm{in}}$ or $^{\mathrm{sa}}$. For each objective user, our proposed model can generate k synthetic $\overline{\mathbf{r}}_u^*, \overline{\mathbf{f}}_u^*$ pairs. Generally, we treat each vector pair as a virtual user u' for the objective user u, named *self-neighbors*. And each virtual user u''s preferences can be formulated as follows:

$$\mathbf{r}_{u'}^* = \overline{\mathbf{r}}_u^* \odot \overline{\mathbf{f}}_u^*, \tag{7}$$

where $\odot$ denotes the element-wise product. Note that the number k of self-neighbors can be tuned for better performance, and we will discuss it in our experiment section. Then we feed the virtual neighbor u''s preference vectors and original user u's preference vector $\mathbf{r}^*$ into the preference filtering module to achieve self-preference $\overline{\mathbf{r}}_u^*$ (Formula (8)), for interest and satisfaction, respectively. Note that we can achieve $\overline{\mathbf{r}}_u^*$ with different operations among original vectors and their virtual neighbors, such as min, max, average, threshold mechanism, network, etc. In our framework, we compute the similarity of vectors and output an average value of *Top3* vectors.

$$\overline{\mathbf{r}}_u^* = \text{Pre-Filtering}\left(\mathbf{r}_u^*, \mathbf{r}_{u_1'}^*, \ldots \mathbf{r}_{u_k'}^*\right). \tag{8}$$

Next, a gated serendipity preference fusion operation is employed for marking each potential candidate item for objective users. First, we give a reminder of *serendipity items:* the items with high relevance but low shopping purpose [16]. With this consideration, intuitively, we can deduce that the items with high satisfaction but low interest should be marked as serendipity items, which means that the objective user would achieve a much better experience after buying them, also introducing additional knowledge for tackling the filter-bubble problem. To obtain the serendipity item set, we set the thresholds θ^* for Interest and Satisfaction, respectively. The item with $r^{\text{sa}} \geq \theta^{\text{sa}}$ but $r^{\text{in}} < \theta^{\text{in}}$ is marked as s_{ui}=1, as a serendipity item. For each user, there is an indicator vector $\mathbf{s}_u$ composed of s_{ui} to mark his serendipity items. In this way, we obtain an intuitive indicator for a user on an item with its interest, satisfaction, and serendipity.

Finally, to relieve the cold-start issue by reducing the sparsity of the user-item matrix, we employ a negative matrix injection method. The idea is that we mark the most impossible items by injecting zeros into the user-item rating matrix $\mathbf{R}$. The reason is that 1) the direct rating deduction from users' preferences is usually tricky and inaccurate, with many uncontrollable factors. 2) In real-world scenarios, the potential items of an objective user make up a small part of thousands or millions of unobserved items, which means that most items in the user-item matrix are meaningless to a specific user, according to the user's preference. Hence, adding zeros for impossible items could solve the sparsity problem greatly.

As a realization of the negative matrix injection, we filter the items with two vectors $\overline{\mathbf{r}}_u^{\text{in}}$ and $\overline{\mathbf{r}}_u^{\text{sa}}$ for user u's negative injections. There are several situations with different values, ($\overline{r}_{ui}^{\text{in}}$ and $\overline{r}_{ui}^{\text{sa}}$, ith is the location index of both vectors). Note that the $\overline{\mathbf{r}}_u^{\text{in}}$ and $\overline{\mathbf{r}}_u^{\text{sa}}$ are computed by Formula (7), and the element could be a value from $[\overline{r}_{ui}^*, 0, ?], 0 < \overline{r}_{ui} \leq 1$. Generally, we design the following gate mechanism to inject $r_{ui}^{\text{h}} = 0$ into enhanced matrix $\mathbf{R}^{\text{h}}$, for all the unrated/unseen items:

- If $\overline{r}_{ui}^{\text{in}} < \theta^{\text{in}}$ and $\overline{r}_{ui}^{\text{sa}} < \theta^{\text{sa}}$, inject $r_{ui}^{\text{h}} = 0$;
- If $\overline{r}_{ui}^{\text{in}} < \theta^{\text{in}}/\overline{r}_{ui}^{\text{sa}} < \theta^{\text{sa}}$, and $\overline{r}_{ui}^{\text{sa}}/\overline{r}_{ui}^{\text{in}} = ?$, inject $r_{ui}^{\text{h}} = 0$;
- Else, we set $r_{ui}^{\text{h}} = r_{ui}$.

Note that we mark s_{ui}=1 for item i that $\overline{r}_{ui}^{\text{in}} < \theta^{\text{in}}$ and $\overline{r}_{ui}^{\text{sa}} > \theta^{\text{sa}}$ as serendipity, and build a serendipity matrix S. Intuitively, the impossible item set for recommendations consists of low-interest and low-satisfaction (below the thresholds) items. Also, we inject 0 s for the items with one unknown preference (?) and one low interest/satisfaction preference. The enhanced user-item matrix $\mathbf{R}^{\text{h}}$ can give more meaningful feedback to indicate users' preferences on items and reduce the unknown feedback, which relieves the cold-start problem. While for the items/users with little or 0 feedback (cold-start items/users in recommender systems) in the original user-item matrix $\mathbf{R}$, our proposed model can inject some zeros as initialized "ratings". It relieves the new user/item cold-start problem. Note that if the original user-item matrix has been updated (utilizes many new r_{ui} to replace ?), we can utilize GS^2-RS again to update $\mathbf{R}^{\text{h}}$, as an improvement for enhancing accuracy.

With $\mathbf{R}^{\text{h}}$ as input (replacing original $\mathbf{R}$), $\overline{\mathbf{R}}^{\text{in}}$, and $\overline{\mathbf{R}}^{\text{sa}}$, we can obtain different types (Top-k, CTR, or next purchase, etc) recommendations $\mathbf{L}$ with different RS models, and additional side information $\overline{\mathbf{R}}^{\text{in}}$, $\overline{\mathbf{R}}^{\text{sa}}$, $\mathbf{S}$. Note that this information is extracted from historical user-item interactions without any additional sources. Meanwhile, they are not necessary for some recommender systems. The recommendation results are formulated as

$$\mathbf{L}= \text{RS model}\left(\mathbf{R}^{\text{h}}, \overline{\mathbf{R}}^{\text{in}}, \overline{\mathbf{R}}^{\text{sa}}, \mathbf{S}\right). \tag{9}$$

Algorithm 1: Generative Self-Serendipity Recommendation System.

```
Require: Original user-item matrix $\mathbf{R}$.
Ensure: Recommendations results $\mathbf{L}$.
 1: Initialization: condition vectors $\mathbf{c}^*$, user latent vector $\mathbf{z}_u$,
    network parameters and threshold $\alpha, \theta$.
Step 1: Preference Extraction
 2: Calculate interest matrix $\mathbf{R}^{\text{in}}$, satisfaction matrix $\mathbf{R}^{\text{sa}}$
    with threshold $\alpha_{ui}$;
 3: Train Twin-CGAN with Formulas (3) and (5) with
    optimization (Section IV-C);
 4: Output generators $\mathcal{G}_{\text{r}}^{\text{in}}, \mathcal{G}_{\text{f}}^{\text{in}}, \mathcal{G}_{\text{r}}^{\text{sa}}, \mathcal{G}_{\text{f}}^{\text{sa}}$;
Step 2: Gated Serendipity Preference Fusion and
  Negative Matrix Injection
 5: for objective user $u$ do
 6:   Ensure self-neighbor number $t$;
 7:   for each self-neighbor $u'$ do
 8:     Generate $\overline{\mathbf{r}}_u^{in}, \overline{\mathbf{f}}_u^{\text{in}}, \overline{\mathbf{r}}_u^{\text{sa}}, \overline{\mathbf{f}}_u^{\text{sa}}$ with $\mathcal{G}_{\text{r}}^{\text{in}}, \mathcal{G}_{\text{f}}^{\text{in}}, \mathcal{G}_{\text{r}}^{\text{sa}}, \mathcal{G}_{\text{f}}^{\text{sa}}$;
 9:     Calculate $\mathbf{r}_{u'}^{\text{in}}, \mathbf{r}_{u'}^{\text{sa}}$ with Formula (7);
10:   end for
11:   Calculate $\overline{\mathbf{r}}_u^{\text{in}}, \overline{\mathbf{r}}_u^{\text{sa}}$ with Formula (8);
12:   Compare $\overline{\mathbf{r}}_u^{\text{in}}$ with $\overline{\mathbf{r}}_u^{\text{sa}}$ element-wised with $\theta^{\text{in}}, \theta^{\text{sa}}$;
13:   Mark the self-serendipity indicator $s_{ui}$;
14:   Inject 0 s to form $\mathbf{r}_{ui}^{\text{h}}$;
15: end for
16: Output $\mathbf{R}^{\text{h}}, \overline{\mathbf{R}}^{\text{in}}, \overline{\mathbf{R}}^{\text{sa}}$ and $\mathbf{S}$;
Step 3: Recommendations
17: Input $\mathbf{R}^{\text{h}}$ instead of $\mathbf{R}$ into SOTA RS models;
18: Input $\overline{\mathbf{R}}^{\text{in}}, \overline{\mathbf{R}}^{\text{sa}}$ and $\mathbf{S}$ as side information into SOTA RS
    models;
19: Calculate $\mathbf{L}$ with Formula (9);
20: return Recommendations results $\mathbf{L}$.
```

In our model, we just employ basic CF models on explicit feedback as RS models. Algorithm 1 describes the complete GS^2-RS model:

C. *Optimizations for GS^2-RS*

Most computations of GS^2-RS occur in the training process of Twin-CGAN. How to optimize this Twin-CGAN is the key to enhancing the GS^2-RS's efficiency. Thus, we propose two optimization strategies for Twin-CGAN, which focus on example augmentation and training.

1) Random Masking for Example Augmentation: For each user u, GS^2-RS obtains one ground truth example ($\mathbf{r}_u^{\text{in}}, \mathbf{r}_u^{\text{sa}}, \mathbf{f}_u$) (Fig. 3) for training Twin-CGAN, which means that only one example to train $\mathcal{G}_{\text{r}}^{\text{in}}, \mathcal{G}_{\text{f}}^{\text{in}}, \mathcal{G}_{\text{r}}^{\text{sa}}, \mathcal{G}_{\text{f}}^{\text{sa}}$. However, the limited number of examples damages the performance of GAN, making them difficult to achieve convergence. As an example augmentation, we utilize a random masking operation to build several examples for one specific user. We randomly mask $\mathbf{r}_u^{\text{in}}, \mathbf{r}_u^{\text{sa}}$ with different $\mathbf{f}_u$. Specifically, we set a Bernoulli distribution Ber(f_{ui}, p_{f}^i) for each position ($f_{ui} = 1$) to decide the value of f_{ui} is 1 or 0, and

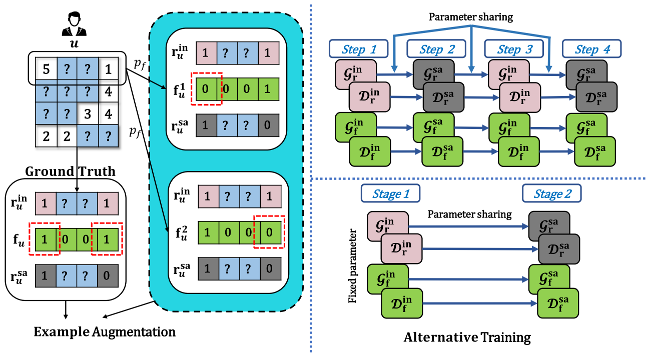


Fig. 5. Optimizations for GS^2-RS. Left: Example augmentation. Right Upper: Step-level alternative training; Right Lower: Stage-level alternative training.

form the masked $\hat{\mathbf{f}}_u$

$$\overline{f}_{ui} = f_{ui} \sim \text{Ber}(p_f^i), \text{if } \ f_{ui} = 1;$$
$$\overline{f}_{ui} = f_{ui}, \text{if } \ f_{ui} = 0. \tag{10}$$

In this way, we build several $\hat{\mathbf{f}}_u$ to form several ground-truth examples $(\mathbf{r}_u^{\text{in}}, \mathbf{r}_u^{\text{sa}}, \hat{\mathbf{f}}_u)$, which augments the examples and accelerates the training. Moreover, the augmented examples enhance the model's robustness by reducing the sensitivity to specific data.

2) Alternative Training for Twin-CGAN: It is widely recognized that adversarial training suffers from the unstable and mode collapse problem, where the generator is easy to over-optimize for a particular discriminator [20]. Twin-CGAN needs to train four pairs of GAN: $\mathcal{G}_{\text{r}}^{\text{in}}$, $\mathcal{G}_{\text{f}}^{\text{in}}$, $\mathcal{G}_{\text{r}}^{\text{sa}}$, $\mathcal{G}_{\text{f}}^{\text{sa}}$, $\mathcal{D}_{\text{r}}^{\text{in}}$, $\mathcal{D}_{\text{f}}^{\text{in}}$, $\mathcal{G}_{\text{r}}^{\text{sa}}$, $\mathcal{G}_{\text{f}}^{\text{sa}}$. If we train each GAN separately, the energy will be wasted, and the efficiency will not be ensured. In this context, we propose an alternative training method for Twin-CGAN, including step-level and stage-level alternative training methods. The motivation is that all the generators/discriminators have the same network structure with the same hyper-parameters. Moreover, the input and output vectors of generators/discriminators have the same dimension, which can be treated as a representation of a specific user's preferences. At last, from our experiment results, we notice that the parameters' distributions are similar between different generators/discriminators. Thus, we propose to copy the parameters for each step of training, from $\mathcal{G}_{\text{r}}^{\text{in}}$, $\mathcal{G}_{\text{f}}^{\text{in}}$, $\mathcal{D}_{\text{r}}^{\text{in}}$, $\mathcal{D}_{\text{f}}^{\text{in}}$, to $\mathcal{G}_{\text{r}}^{\text{sa}}$, $\mathcal{G}_{\text{f}}^{\text{sa}}$, $\mathcal{G}_{\text{r}}^{\text{sa}}$, $\mathcal{G}_{\text{f}}^{\text{sa}}$, alternatively, as the step-level method. Or we can train $\mathcal{G}_{\text{r}}^{\text{in}}$, $\mathcal{G}_{\text{f}}^{\text{in}}$, $\mathcal{D}_{\text{r}}^{\text{in}}$, $\mathcal{D}_{\text{f}}^{\text{in}}$ in one batch, and copy their parameters as the initialization to $\mathcal{G}_{\text{r}}^{\text{sa}}$, $\mathcal{G}_{\text{f}}^{\text{sa}}$, $\mathcal{G}_{\text{r}}^{\text{sa}}$, $\mathcal{G}_{\text{f}}^{\text{sa}}$, as the stage level method. With this method, we can speed up the training process of Twin-CGAN. For small datasets, the step-level method can achieve superior performance, while for large datasets with many batches, the stage-level method is a proper choice. Fig. 5 exhibits the two optimizations:

D. Model Enhancement Analysis

This section introduces how our proposed model solves the FB problem and enhances the SOTA recommenders.

1) Enhancing Personalizations for FB Problem: Personalized Recommendation is an important factor in recommender systems because a boring, homogeneous recommendation is not expected for each user. For enhancing personalized recommendations, existing recommenders can employ users' preferences predicted by GS^2-RS (interest, satisfaction, and serendipity) to rank recommendation results $\mathbf{L}$. With interest $\overline{\mathbf{R}}^{\text{in}}$, satisfaction $\overline{\mathbf{R}}^{\text{sa}}$ and self-serendipity $\mathbf{S}$, we can make a fine-grained user profiling for personalized recommendations, and replace Formula (9) as follows:

$$\mathbf{L}\text{= RS model}\left(\mathbf{R}^{\text{h}}\right) + \text{Ranking model}\left(\overline{\mathbf{R}}^{\text{in}}, \overline{\mathbf{R}}^{\text{sa}}, \mathbf{S}\right). \tag{11}$$

Moreover, we can check the distribution of interest and satisfaction items in historical records to distinguish the objective user's preference distribution. Then we rerank the recommendation list in accordance with this user's preference distribution. Besides, the self-serendipity item should also be considered: we could level up the importance of self-serendipity items in $\mathbf{L}$ for enhancing the diversity and serendipity of recommenders. With different ranking models based on interest $\overline{\mathbf{R}}^{\text{in}}$, satisfaction $\overline{\mathbf{R}}^{\text{sa}}$ and self-serendipity $\mathbf{S}$, we can add the information entropy gain of recommendations (Formula (1)) thus the filter-bubble problem can be adequately solved.

2) Enhancing CF/MF/NN Based Recommenders: Existing recommender systems usually consist of three specific categories: collaborative filtering-based RS (CF models), matrix factorization-based RS (MF models), and neural network-based RS (NN models). GS^2-RS can enhance these recommender systems for the following reasons: For enhancing CF models, GS^2-RS could give more valuable feedback in $\mathbf{R}^{\text{h}}$, (0 s that injected in Negative Matrix Injection), which can help the model to compute the distance between different users/items, and select more accurate users' to filter the items.

For MF models, we employ WRMF or other matrix factorization to learn users'/items' latent vectors from user-item matrix $\mathbf{R}$ and then make recommendations. Note that all the existing matrix factorization algorithms' performance is affected dramatically by the matrix's sparsity, while our proposed model GS^2-RS can relieve the sparsity problem by replacing $\mathbf{R}$ with $\mathbf{R}^{\text{h}}$. Moreover, the 0 s in $\mathbf{R}^{\text{h}}$ also can be treated as ratings, which can restrict the learned latent users'/items' vectors for accurate performance. Note that MF models are usually employed as the preprocessing for NN models to get the input of the neural network framework. Moreover, the input is vital for NN models. Indeed, our proposed model can enhance NN models by offering user/item embeddings as the input. We will give a discussion in the experimental section.

V. Validations

This section validates our proposed framework. We conduct extensive experiments on four real-world datasets to answer the following research questions:

RQ1: How does our proposed GS^2-RS enhance the overall recommendation performance over the state-of-the-art models?

TABLE I
DATASETS STATISTICS

Datasets	*#Users*	*#Items*	*#feedback*	Sparsity
Movielens	6,040	3,952	1,000,209	95.81%
Amazon	6,403,006	1,660,119	14,771,988	99.99%
Yelp	43,873	11,537	229,907	98.70%
Book-Crossing	278,858	271,379	1,149,780	99.99%

RQ2: What is the effect for each module in GS^2-RS for the recommendation? What is the effect of the optimization method for GS^2-RS?

RQ3: How does GS^2-RS solve the cold-start and filter-bubble problems in a unified framework? Can it solve both problems by reducing sparsity or adding information entropy of recommendations?

RQ4: How do the parameters and thresholds affect GS^2-RS's performance?

A. Experimental Details

1) Dataset Details: To evaluate the performance of GS^2-RS, we utilize four datasets: Movielens,[1] Amazon,[2] Yelp[3] and Book-Crossing,[4] which are exclusively published by [21]. Movielens is the classical benchmark dataset that is widely used to validate recommender systems. Amazon and Yelp represent recommendation tasks for different domains (E-commerce and Location recommendation, respectively). Book-crossing is composed of the ratings extracted from a book-sharing website. These datasets contain thousands or millions of explicit feedback (ratings ranging from 1 to 5). Note that all these datasets are extremely sparse (over 95%). Details are indicated in Table I. The code is released: https://github.com/CodeAnonymize/tkde2023.

2) Baseline Methods: To validate GS^2-RS, we select several classic RS models and SOTA RS models as the validations: *CF* [22]. This method is a traditional recommender widely applied to the user-item rating matrix. It computes the similarity between different users/items, finding the nearest neighbors and making recommendations according to these neighbors. *WMF* [2], [23]. This explicit feedback prediction model represents both users and items as low-dimension latent vectors. This model is optimized to predict the observed rating for each user-item pair, which considers the confidence for each rating. *NCF* [9]: This is a general neural-network-based recommendation framework, which uses Multi-Layer Perceptron (MLP), Generalised Matrix Factorisation (GMF), and Neural Matrix Factorisation (NeuMF) for collaborative filtering. *JoVA* [24]: This method is an ensemble of two VAEs to capture multiple correlations between users and items simultaneously for the recommendation. This method does not consider Cold-Start and Filter-Bubble problems. *AR-CF* [3]: This method is a GAN-based CF model, which is applied directly to generating ratings for tackling the Cold-Start problem. *CSII* [16]: a SOTA serendipity-based method with a modified MF algorithm, which introduces the serendipity as a metric to solve the Filter-Bubble problem. *GAZRec* [25]: a generative adversarial zero-shot learning framework for cold-start issue in news recommendation. We refine this model to suit the datasets we employ. *VELF*: [26]: a general variational embedding learning framework for alleviating the severe cold-start problem in CTR prediction. We refine this model to suit the datasets we employ.

3) Evaluation Metrics: Because GS^2-RS can solve Cold-start and Filter-bubble problems in a unified recommendation framework, we comprehensively evaluate our method from three aspects: Recommendation, Effect on CS problem, and Effect on FB problem with different metrics:

Recommendation: We employ Precision, Recall, Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR) to validate overall recommendation performance. We define $\mathbf{I}^{\text{rec}}$ is the recommendation item list, including k items; $\mathbf{I}^{\text{real}}$ is the ground-truth of the test set

$$\begin{aligned} Pre@k &= \#num(\mathbf{I}^{\text{rec}} \cap \mathbf{I}^{\text{real}})/\#num(\mathbf{I}^{\text{rec}}). \\ Rec@k &= \#num(\mathbf{I}^{\text{rec}} \cap \mathbf{I}^{\text{real}})/\#num(\mathbf{I}^{\text{real}}). \\ NDCG@k &= DCG@k/iDCG@k. \\ MRR@k &= \frac{1}{k}\sum_{c=1}^{k}\frac{1}{pos_{i_c}}. \\ MISI &= \frac{\sum_u num(\text{items with feedback but injected 0})}{\sum_u num(\text{items injected 0})}. \end{aligned} \tag{12}$$

Specifically, DCG is the rank score of the recommendation list, while iDCG is the ideal rank score. pos_i is item i's ideal position in $\mathbf{I}^{\text{real}}$. MISI is the metric for the injection estimation for GS^2-RS.

Effect on CS Problem: We utilize the Exposure Ratio (ER) as the metric for the CS problem. Formally, ER is computed by B/A, where B is the number of cold-start items which are exposed to at least one user, and A is the number of the entire cold-start items. Using the same definitions before, we rewrite ER as

$$ER@k = \#num(\mathbf{I}^{\text{rec}} \cap \mathbf{I}^{\text{cs}})/\#num(\mathbf{I}^{\text{cs}}), \tag{13}$$

where I^{cs} is composed of cold-start items.

Effect on FB Problem: we utilize diversity (DI) and serendipity (SE) [16] as metrics. DI is a direct metric to evaluate the information entropy gain of a recommendation. SE is a metric to represent how many high-satisfaction but low-interest items are recommended, formulated as

$$\begin{aligned} DI@k &= \#category(\mathbf{I}^{\text{rec}})/\#num(\mathbf{I}^{\text{rec}}). \\ SE@k &= \#num(\mathbf{I}^{\text{rec}} \cap \mathbf{I}^{\text{real}} \cap (\mathbf{I}^{\text{sa}} - \mathbf{I}^{\text{in}}))/\#num(\mathbf{I}^{\text{real}}). \end{aligned} \tag{14}$$

4) Experimental Settings: We implement GS^2-RS based on Pytorch accelerated by NVIDIA RTX 3090 GPU. In experiments, we first use Grid search and 5-fold cross-validation to find the best parameters and then use the Adam optimizer

[1]http://grouplens.org/datasets/movielens
[2]http://jmcauley.ucsd.edu/data/amazon/links.html
[3]http://www.yelp.com
[4]http://www.bookcrossing.com

TABLE II
THE OVERALL PERFORMANCE

Datasets	Movielens (MISI=0.0013%)				Amazon (MISI=0.0001%)			
Metrics	*Pre.@10*	*Rec.@10*	*NDCG@10*	*MRR*	*Pre.@10*	*Rec.@10*	*NDCG@10*	*MRR@10*
CF	0.0193	0.0893	0.2210	0.4056	0.0088	0.0049	0.1129	0.3656
WMF	0.1541	0.1644	0.3059	0.4731*	0.1032	0.1321	0.2048	0.4111
NCF	0.1873	0.1831	0.2710	0.3594	0.1321	0.1224	0.2321	0.3321
JoVA	0.2010*	0.2001*	0.3010	0.4687	0.1177	0.1331	0.3015	0.4014
AR-CF	0.1707	0.1127	0.3971*	0.4332	0.1421	0.1644	0.4015*	0.4233*
CSII	0.1635	0.1721	0.3641	0.3722	0.1333	0.1531	0.3312	0.3613
GAZRec	0.1933	0.1938	0.3961	0.4721	0.1411	0.1599	0.3991	0.4231
VELF	0.2009	0.2001*	0.3962	0.4729	0.1513*	0.1655*	0.4011	0.4239
GS^2-RS	**0.2230**	**0.2006**	**0.4449**	**0.4837**	**0.1534**	**0.1756**	**0.4333**	**0.5210**
Perform.+	*10.94%*	*0.24%*	*12.03%*	*2.24%*	*1.38%*	*6.10%*	*7.92%*	*23.08%*
Datasets	Yelp (MISI=0.0007%)				Book-Crossing (MISI=0.0001%)			
Metrics	*Pre.@10*	*Rec.@10*	*NDCG@10*	*MRR@10*	*Pre.@10*	*Rec.@10*	*NDCG@10*	*MRR@10*
CF	0.0188	0.0883	0.2147	0.3743	0.0123	0.0102	0.1312	0.3312
WMF	0.1423	0.1522	0.3143	0.3822	0.1002	0.1134	0.1873	0.3710
NCF	0.1631	0.1723	0.2532	0.3132	0.1112	0.1321	0.2411	0.3012
JoVA	0.1997*	0.1766*	0.3252	0.4126*	0.1201	0.1411	0.3152	0.3822
AR-CF	0.1813	0.1644	0.3455*	0.3911	0.1332*	0.1466*	0.3465*	0.3911*
CSII	0.1799	0.1742	0.3401	0.3822	0.1301	0.1450	0.3382	0.3702
GAZRec	0.1988	0.1745	0.3362	0.4098	0.1303	0.1432	0.3391	0.3871
VELF	0.1976	0.1755	0.3391	0.4121	0.1312	0.1457	0.3411	0.3891
GS^2-RS	**0.2112**	**0.1978**	**0.3821**	**0.4432**	**0.1523**	**0.1666**	**0.4002**	**0.4213**
Perform.+	*5.75%*	*12.00%*	*10.59%*	*7.41%*	*14.34%*	*13.64%*	*15.50%*	*7.72%*

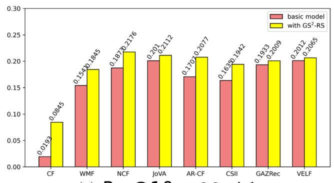
The bold font denotes the winner, and*denotes the second winner in that column. Perform.+ denotes the performance gain percentage over the second winner model, which is significant with t-test p<0.05. Note that MISI is just for GS2-RS.

to optimize our loss function, where the mini-batch size is set to 128. The initial learning rate is set to 0.001. In our proposed GS^2-RS, we set the hyper-parameters: the thresholds are set: $\alpha_u = \sum_i r_{ui}/\#num(r_{ui})$, $\theta^{\text{in}} = \theta^{\text{sa}} = 0.5$. In Twin-CGAN, we form the generator/discriminator with N=3 attention blocks. And the latent dimension for $\mathbf{c}_u^*$ and $\mathbf{z}_u$ is 128. We generate 2 virtual neighbors for each user as initialization. To avoid overfitting, the dropout regularization method with drop ratios 0.08/0.1/0.08/0.1 is utilized for four datasets. For the optimization of GS^2-RS, according to the different scales of datasets, we sample 5 masked examples from Formula (10) for Movielens and Yelp with step-level alternative training and 3 for Amazon and Book-Crossing with stage-level alternative training. To reduce the sample bias, we employ normal random negative sampling for our proposed method, but the reported negative sampling of the baselines (we do not utilize the designed negative sampling to achieve a performance improvement). For the settings of baseline methods, we try our best to tune the hyper-parameters reported in their publications to obtain optimal performance.

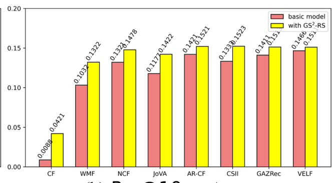
B. *Overall Performance Validation (*RQ1*)*

The overall performance across four datasets is reported in Table II. From the reported MISI, we notice that the 0 injection is accurate and relative to the sparsity of different datasets, which validate the estimation accuracy of GS^2-RS' 0 injection module. Note that in smaller datasets (Movielens and Yelp), JoVA performs better than other baselines with its two VAEs to capture the users' preference for recommendations. In Movielens, JoVA obtains a competitive *Rec@10* to our proposed GS^2-RS. However, on the other three datasets, GS^2-RS outperforms JoVA for an average 12% on *Rec@10*. The reason is that most users' preferences are stable and fixed, which benefits the generative model like JoVA, VELF, GAZRec, and GS^2-RS. While in other datasets, users' preferences are usually dynamic (e.g., in Yelp, users' preferences are shifted with time because they may go to different places at different time.), which is difficult for traditional VAEs to converge. GS^2-RS is not only a generative model but a unified recommender framework that utilizes GAN and the idea of CF. Thus it achieves a better performance on the dynamic-preference dataset than JoVA. AR-CF and VELF obtain a competitive performance in Amazon and Book-Crossing to our proposed GS^2-RS. Both methods utilize GAN to generate users' ratings to build their neighbors, proper for massive and sparse datasets. However, they ignore that users' preferences are the core reason for predicting ratings. Only generating ratings may lead to vital overfitting.

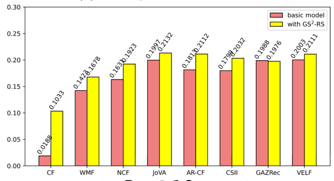
Moreover, to fully answer RQ1, we set GS^2-RS as the preprocessing for the state-of-the-art recommendation models. Specifically, we input the original user-item matrix $\mathbf{R}$ in GS^2-RS and use our model's output $\mathbf{R}^{\text{h}}$ as the other models' input to validate whether our method can enhance existing models. The results are reported in Figs. 6 and 7. Note that we hope these experiments reflect the accuracy and ranking gain, so we choose *Pre@10* and *NDCG@10* as metrics. The results show that our proposed model can enhance the SOTA models' Precision and NDCG performance on four datasets without changing their settings or parameters, especially on the basic CF models. The reason is that the CF model is sensitive to input sparsity while GS^2-RS can generate users' preferences and use negative injections to relieve sparsity issues. Besides, we notice that GS^2-RS benefits NCF more than other baselines. As we explained in Section IV-D2, NCF is a neural network-based model, and the quality of its input



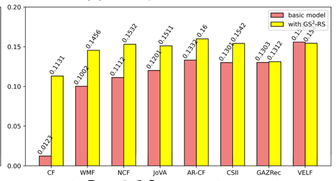
(a) *Pre*@10 on Movielens



(b) *Pre*@10 on Amazon

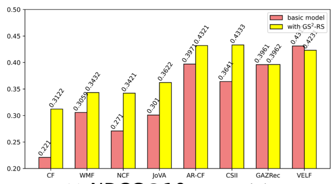


(c) *Pre*@10 on Yelp

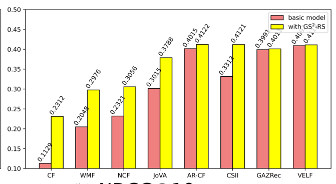


(d) *Pre*@10 on Book-Crossing

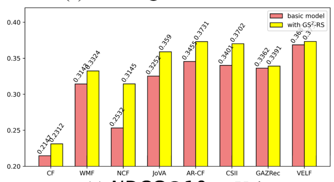
Fig. 6. Enhancing recommenders' performance (*Precision*) as Preprocessing.



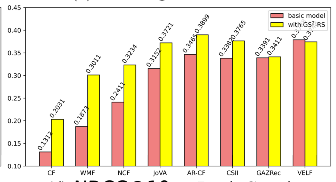
(a) *NDCG*@10 on Movielens



(b) *NDCG*@10 on Amazon



(c) *NDCG*@10 on Yelp



(d) *NDCG*@10 on Book-Crossing

Fig. 7. Enhancing recommenders' performance (*NDCG*) as Preprocessing.

decides its performance. This method employs GMF to tackle the user-item rating matrix achieving user/item latent vectors as users' preferences/item's attributes. With GS^2-RS as preprocessing, $\mathbf{R}^{\mathrm{h}}$ offers more explicit information than the original $\mathbf{R}$ as NCF's input. Thus NCF's performance can be improved. For CSII, a serendipity-based MF model, our proposed method can extract serendipity items from historical ratings. With a similar definition of serendipity item, CSII can extend the result of GS^2-RS and improve its performance.

Besides, we notice that two recent baselines VELF and GAZRec outperform other baselines in Table II, but are slightly weak compared with GS^2-RS. The reason is that both methods' core idea is to utilize a generative way (generate feedbacks) to enhance the performance of recommender systems, which is similar to our proposed methods (generate preferences). When it comes to Figs. 6 and 7, we find that utilizing GS^2-RS for VELF and GAZRec may weaken the performance. This phenomenon may be caused by the over-generative ability (both preprocessing and recommendation models are generative ones, which may introduce more bias and damage the accuracy). But our proposed method does increase the other baselines' performance as the preprocessing method, which proves its effectiveness.

C. *Ablation Study (*RQ2*)*

To validate the effect of each module in GS^2-RS, we build several variants of GS^2-RS: 1) GS^2-RS-**R**: a variant without the preference extraction module. It transfers user-item matrix R into binarization as $\mathbf{R}^{\mathrm{in}}$ and $\mathbf{R}^{\mathrm{sa}}$. 2) GS^2-RS-in: a variant without satisfaction generative part. It only uses users' Interests from I-CGAN as their preferences. 3) GS^2-RS-sa: a variant without interest generative part. It only uses users' Satisfaction from S-CGAN as their preferences. 4) GS^2-RS-*no***f**: a variant without the indicator vector **f**. 5) GS^2-RS-*no***NMI**: a variant without negative matrix injection. It utilizes a linear combination of Interest and Satisfaction as users' preference and input them into **R**. We conduct these variants on two datasets of different scales: Movielens and Amazon. The results are reported in Table III. We notice that the model's performance is damaged by removing each module. Specifically, GS^2-RS-**R** performs worst, and GS^2-RS-in achieves the best performance among these variants, which is in line with the intuition that users are willing to buy the items they are interested in. Using Interest as users' preference is a common method employed by some recommender systems. They transfer ratings into binarization, like GS^2-RS-**R**. Nevertheless, in our experiment, this variant can not predict fine-grained users' preferences, which results in poor performance. Besides Interest, by introducing Satisfaction, Indicator vector, and negative matrix injection into our framework, the model's performance can be improved, validating these modules' effect for the recommendation.

To validate the effect of our optimization method on GS^2-RS, we build two variants of GS^2-RS: GS^2-RS-*no***EA**: a variant without example augmentation, which means that there is only one example for each user as ground truth. GS^2-RS-*no***AT**: a variant that trains the I-CGAN and S-CGAN without alternative training. Note that for GS^2-RS, we use step-level alternative training on small datasets (Movielens and Yelp) and stage-level on large ones (Amazon and Book-Crossing). The results are reported in Table IV. We notice that GS^2-RS-*no***AT** achieves competitive performance with GS^2-RS, but it uses more time for training. Moreover, the performance gap between GS^2-RS-*no***EA** and GS^2-RS indicates that the example augmentation can significantly enhance our model's generalization ability. Note that we can randomly mask indicator vector $\mathbf{f}_u$ to build many examples for a specific user u. However, we notice that performance gain becomes small after the example number increases to 8. So we fix the example augmentation number with 3 for small datasets and 5 for large datasets to achieve a trade-off between efficiency and accuracy.

D. *Effect on Solving Cold-Start Problem (* RQ3*)*

Cold-start problem is challenging because cold-start users/items are seldom to be observed and exposed to users. We hope to explore whether our proposed GS^2-RS offers these cold-start users/items more opportunities. We employ the Exposure Ratio (*ER*@k) to evaluate the performance for solving the cold-start problem. As the ground truth, we define the most infrequent items with little explicit feedback as the Cold-Start items in the training dataset. Note that because we utilize only the

TABLE III
THE ABLATION STUDY FOR EACH MODULE IN GS^2-RS

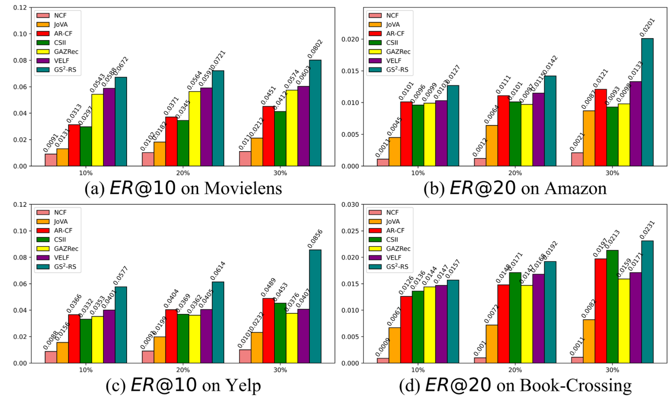
Datasets	Movielens				Amazon			
Metrics	*Pre.*@10	*Rec.*@10	*NDCG*@10	*MRR*	*Pre.*@10	*Rec.*@10	*NDCG*@10	*MRR*@10
GS^2-RS-**R**	0.1523 ↓ 31.7%	0.1342 ↓ 33.1%	0.3892 ↓ 12.5%	0.4003 ↓ 17.8%	0.1212 ↓ 21.0%	0.1431 ↓ 18.5%	0.3522 ↓ 18.7%	0.4321 ↓ 17.1%
GS^2-RS-in	0.1990 ↓ 10.7%	0.1843 ↓ 8.1%	0.4002 ↓ 10.0%	0.4323 ↓ 11.3%	0.1432 ↓ 6.7%	0.1534 ↓ 12.6%	0.3923 ↓ 9.5%	0.4834 ↓ 7.2%
GS^2-RS-sa	0.1833 ↓ 17.8%	0.1632 ↓ 18.6%	0.4102 ↓ 7.8%	0.4384 ↓ 10.0%	0.1410 ↓ 8.1%	0.1512 ↓ 13.9%	0.4101 ↓ 5.4%	0.4921 ↓ 5.5%
GS^2-RS-*no***f**	0.1611 ↓ 27.8%	0.1762 ↓ 12.2%	0.4112 ↓ 7.6%	0.4212 ↓ 13.6%	0.1332 ↓ 13.2%	0.1532 ↓ 12.8%	0.4011 ↓ 7.4%	0.4623 ↓ 11.3%
GS^2-RS-*no***NMI**	0.1521 ↓ 31.8%	0.1422 ↓ 29.1%	0.3789 ↓ 14.8%	0.3822 ↓ 21.6%	0.1273 ↓ 17.0%	0.1388 ↓ 21.0%	0.3743 ↓ 13.6%	0.4452 ↓ 14.5%
GS^2-RS	**0.2230**	**0.2006**	**0.4449**	**0.4837**	**0.1534**	**0.1756**	**0.4333**	**0.5210**

The bold font denotes the winner, and ↓ denotes the performance decrease compared with the winner.

TABLE IV
THE ABLATION STUDY FOR OPTIMIZATION METHODS

Datasets	Movielens			Amazon		
Metrics	Pre.@10	NDCG@10	T-time	Pre.@10	NDCG@10	T-time
GS^2-RS-*no***EA**	0.1882	0.4123	47	0.1312	0.4266	142
GS^2-RS-*no***AT**	0.2228	0.4451	154	0.1532	0.4344	348
GS^2-RS	0.2230	0.4449	89	0.1534	0.4333	213
Datasets	Yelp			Book-Crossing		
Metrics	Pre.@10	NDCG@10	T-time	Pre.@10	NDCG@10	T-time
GS^2-RS-*no***EA**	0.1982	0.3423	53	0.1399	0.3789	163
GS^2-RS-*no***AT**	0.2110	0.3811	201	0.1521	0.4010	934
GS^2-RS	0.2112	0.3821	104	0.1523	0.4002	341

T-time is the training time for GS2-RS, where most training time is spent on Twin-CGAN.



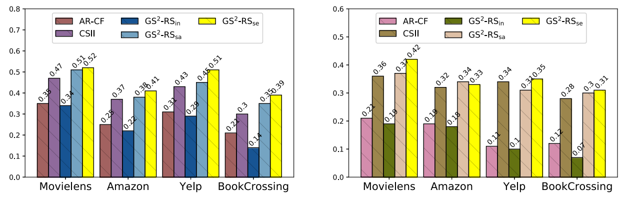
(a) $ER@10$ on Movielens (b) $ER@20$ on Amazon
(c) $ER@10$ on Yelp (d) $ER@20$ on Book-Crossing

Fig. 8. Exposure ratio for Cold-Start items in recommendations.

user-item rating matrix as input, Cold-Start users are the same as Cold-Start items from a model's perspective. Specifically, we employ NCF, JoVA, CSII, AR-CF, GAZRec and VELF as baselines to compare with GS^2-RS, as reported in Fig. 8. $T\%$ denotes the bottom percentage of items that interact with users as cold-start items. We define different $T\%$ for small datasets and large datasets, respectively. We observe that NCF and JoVA have difficulty solving the cold-start problem, while AR-CF, CSII, GAZRec, VELF, and GS^2-RS outperform them considerably, especially on large datasets (Amazon and Book-Crossing). Meanwhile, because our proposed model generates users' preferences but not ratings, GS^2-RS performs better than AR-CF with solid generalization ability. Besides, jointly considered with results in Figs. 6 and 7, NCF and JoVA achieve competitive performance in accuracy but a bad ability to solve the CS problem. The reason is that these two methods are too strong to predict users' preferences, which leads to a sort of "overfitting". These methods pay more attention to items with more ratings, damaging Cold-Start items' chances of being recommended. In contrast, AR-CF and CSII achieve better ER over NCF and JoVA because these methods introduce indeterminacy to predict users' preferences, avoiding overfitting. This experiment demonstrates the effectiveness of our proposed GS^2-RS in solving cold-start issues with a stable accuracy performance.

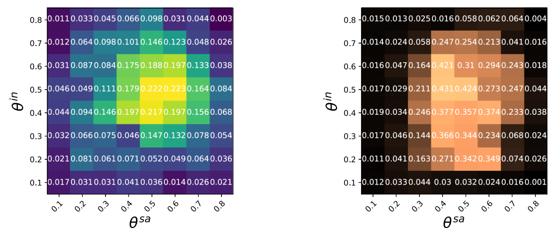
E. *Effect on Solving Filter-Bubble Problem (* RQ3*)*

Filter-bubble problem is difficult because most models recommend similar items to users according to their historical feedback without providing more useful information (information entropy gain). In GS^2-RS, we can rerank the recommendation list in accordance with objective users' historical interest, satisfaction, and serendipity distributions to achieve the trade-off between accuracy and diversity. We utilize different ranking methods: GS^2-RS$_{\text{in}}$, $_{\text{sa}}$, $_{\text{se}}$: ranking the list with descending Interest, Satisfaction, and Serendipity. Note that DI@10 denotes the information entropy gain of the recommendation, and SE@10 denotes the proportion of serendipity items among recommended items. From DI and SE results reported in Fig. 9, we observe that 1) GS^2-RS$_{\text{se}}$ achieves the best diversity and serendipity significantly among the baselines and its variants, especially on sparse datasets. 2) GS^2-RS$_{\text{sa}}$ and CSII achieve second-best



(a) Diversity (DI) @10 (b) Serendipity (SE) @10

Fig. 9. Diversity/Serendipity for the FB problem.



(a) Precision ($Pre.$@10) (b) Serendipity (SE@10)

Fig. 10. Effect of threshold θ on accuracy ($Pre.$@10) and serendipity (SE@10) (better viewed in color mode). The lighter part denotes better performance.

diversity and serendipity compared with other baselines. GS^2-RS$_{\text{in}}$ focuses on users' Interests, which is similar to the existing methods, like AR-CF does, thus it performs worst in solving the FB problem. To sum up, our proposed method can achieve a high diversity and serendipity with a stable accuracy, greatly alleviating the Filter-Bubble problem.

*F. Parameters Analysis (*RQ4*)*

As a reminder, our proposed GS^2-RS' contribution is its ability to predict fine-grained preferences for a recommendation. Hence we hope to explore the effect of the preference threshold. Specifically, we validate the effect of θ, as reported in Fig. 10. Note that we set θ^{in}, θ^{sa} respectively for validation, ranging (0.0, 1,0) with step 0.1, and report the Precision ($Pre.$@10) and Serendipity (SE@10), respectively, We observe that GS^2-RS achieves the best performance at $\theta^{\text{in}} = \theta^{\text{sa}} = 0.5$. When both θ descend to 0, GS^2-RS cannot filter any items, which fades to a basic GAN-based model. When both θ ascend to 1, GS^2-RS drops every candidate item and damages its performance. Besides, we notice that the higher performance occurs in the up-right part of the performance heatmap. This phenomenon indicates that using a gate mechanism with thresholds on users' preferences before recommendation benefits the models' performance.

We also explore the number N of virtual neighbors' effect. Note that we generate several virtual users' preferences and apply simple user-based CF on these virtual users and the objective user. As reported in Fig. 11, when N=2. our model achieves the best trade-off between accuracy and serendipity. And the performance of both Pre@10 and SE@10 are descending when the virtual neighbor's number increases on different datasets. It is intuitive that generating too many virtual neighbors may result in overfitting and hurt the serendipity, while only 1 virtual

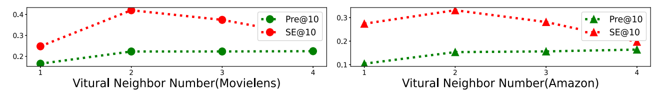


Fig. 11. Effect of virtual neighbor numbers in GS^2-RS.

neighbor could not form useful collaborative signals and hurt the accuracy.

VI. Related Work

This section briefly introduces the recent development of Cold-start and Filter-bubble problems in recommendations.

A. Cold-Start Problem in Recommendations

There are abundant models proposed for relieving the cold-start problem in recent years [27], [28], [29], [30], [31], [32], [33]. [27] proposed a tagging algorithm to tag unobserved items for solving the cold-start issue. [28] built an end-to-end meta-learning framework for sequential recommenders with cold-start users. [29] proposed a pre-trained network modulation and task adaptation approach for cold-start users. [30] utilized transfer learning and meta-learning for tackling cold-start recommendations with cross-domain information. [31] extracted multifaceted meaningful semantics on HINs as multi-views for both users and items, then combined them for recommendations. [32] learned a unified representation with side information and feedback for cold-start users to improve the recommendation models. [33] alleviated the cold-start issue with a follow-the-meta-leader algorithm to learn stable online gradients. [13] utilized cross-domain information to reduce data sparsity for the cold-start problem and enhanced the SOTA recommendation performance. Meanwhile, some researchers explore the usage of GAN [19] for the cold-start problem: [3] generated virtual neighbors for objective users and made accurate recommendations by reducing cold-start items. [34] generated users' embedding for recommendation and improved *NDCG*@100 significantly. To sum up, the above work tries to solve cold-start problems with different models (transfer learning, meta-learning, etc.). However, these frameworks usually utilize complicated frameworks and side information to enhance existing models. Unlike these methods, our proposed method generates users' fine-grained preferences from only the user-item rating matrix, enriching the information of the historical feedback to solve the CS problem.

B. Filter-Bubble Problem in Recommendations

Most researchers treat Filter-Bubble problem as a natural dilemma [7], [17], [35], [36], [37], [38]. Some researchers [17], [22] tried to add the diversity of recommendation list. [35] investigated how different algorithmic strategies affect filter bubble formation and concluded that content-based and collaborative-filtering recommenders result in markedly different filter bubbles. [36] explored the filter bubbles' effect on link prediction task in graphs. [37] introduced a mathematical framework to assess the impact of filter bubbles and indicated that the interference may increase user polarization. [38] devised an echo chamber-aware friend recommendation approach that learns users and echo chamber representations from the shared content and past users' and communities' interactions. [7] developed algorithms to optimize the performative risk with better sample efficiency than generic methods for tackling filter-bubble and echo-chamber problems. [4] adapted users' novelty preferences into recommendations, which added diversity for relieving the filter-bubble problem. Recently, the idea of serendipity has been proposed to solve the filter-bubble problem by offering novel, diverse and high-satisfaction recommendations. [6] gave general explanations about why serendipity items could work for handling filter-bubble situations. [16] proposed a matrix factorization-based model for enhancing serendipity for superior recommendations. In summary, existing models for solving the filter bubble problem focus on learning proper, dynamic user preferences instead of stable ones for recommendation or introducing novel metrics to modify the recommenders. However, it is still challenging to solve cold-start and filter-bubble problems simultaneously. As an improvement, our proposed GS^2-RS is a unified, practical framework that generates fine-grained preferences to solve CS and FB problems for enhancing recommendations.

VII. Concluding Remarks

This work focuses on solving Cold-Start and Filter-Bubble problems in recommender systems. We proposed GS^2-RS, a unified generative framework for addressing CS and FB problems, without any side information. We explored the insights of both situations and devised our model to predict the fine-grained user preferences hidden in historical feedback. Besides, we extended our framework as a preprocess for existing recommenders. From empirical experiments on public datasets, we demonstrated that GS^2-RS significantly advances accuracy, diversity, and serendipity compared to SOTA RS models, which proves its effectiveness in dealing with both CS and FB problems. In the future, we plan to explore extending GS^2-RS to multi-media recommendation scenarios, incorporating images (with graph convolutional neural network), social networks (with knowledge graph), or context (with natural language processing).

References

[1] L. Zou et al., "Neural interactive collaborative filtering," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, China, 2020, pp. 749–758.

[2] J. Chen et al., "Fast adaptively weighted matrix factorization for recommendation with implicit feedback," in *Proc. 34th AAAI Conf. Artif. Intell., 32nd Innov. Appl. Artif. Intell. Conf., 10th AAAI Symp. Educ. Adv. Artif. Intell.*, New York, NY, USA, AAAI Press, 2020, pp. 3470–3477.

[3] D. Chae, J. Kim, D. H. Chau, and S. Kim, "AR-CF: Augmenting virtual users and items in collaborative filtering for addressing cold-start problems," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, China, 2020, pp. 1251–1260.

[4] K. Kapoor, V. Kumar, L. G. Terveen, J. A. Konstan, and P. R. Schrater, ""I like to explore sometimes": Adapting to dynamic user novelty preferences," in *Proc. 9th ACM Conf. Recommender Syst.*, Vienna, Austria, 2015, pp. 19–26.

[5] V. Feldman, R. Frostig, and M. Hardt, "The advantages of multiple classes for reducing overfitting from test set reuse," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, California, USA, PMLR, 2019, pp. 1892–1900.

[6] R. J. Ziarani and R. Ravanmehr, "Serendipity in recommender systems: A systematic literature review," *J. Comput. Sci. Technol.*, vol. 36, no. 2, pp. 375–396, 2021.

[7] J. Miller, J. C. Perdomo, and T. Zrnic, "Outside the echo chamber: Optimizing the performative risk," in *Proc. 38th Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 7710–7720.

[8] L. Hou, X. Pan, K. Liu, Z. Yang, J. Liu, and T. Zhou, "Information cocoons in online navigation," 2021, arXiv:2109.06589. [Online]. Available: https://arxiv.org/abs/2109.06589

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, Perth, Australia, 2017, pp. 173–182.

[10] Y. Xu, Y. Yang, J. Han, E. Wang, J. Ming, and H. Xiong, "Slanderous user detection with modified recurrent neural networks in recommender system," *Inf. Sci.*, vol. 505, pp. 265–281, 2019.

[11] S. Zhang, H. Yin, T. Chen, Q. V. H. Nguyen, Z. Huang, and L. Cui, "GCN-based user representation learning for unifying robust recommendation and fraudster detection," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, China, 2020, pp. 689–698.

[12] D. Kim and B. Suh, "Enhancing VAEs for collaborative filtering: Flexible priors & gating mechanisms," in *Proc. 13th ACM Conf. Recommender Syst.*, Copenhagen, Denmark, 2019, pp. 403–407.

[13] Y. Bi, L. Song, M. Yao, Z. Wu, J. Wang, and J. Xiao, "A heterogeneous information network based cross domain insurance recommendation system for cold start users," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, China, 2020, pp. 2211–2220.

[14] B. Fu, W. Zhang, G. Hu, X. Dai, S. Huang, and J. Chen, "Dual side deep context-aware modulation for social recommendation," in *Proc. Web Conf.*, Ljubljana, Slovenia, 2021, pp. 2524–2534.

[15] X. Wang, I. Ounis, and C. Macdonald, "Leveraging review properties for effective recommendation," in *Proc. Web Conf.*, Ljubljana, Slovenia, 2021, pp. 2209–2219.

[16] Y. Yang, Y. Xu, E. Wang, J. Han, and Z. Yu, "Improving existing collaborative filtering recommendations via serendipity-based algorithm," *IEEE Trans. Multimedia*, vol. 20, no. 7, pp. 1888–1900, Jul. 2018.

[17] L. Burbach, J. Nakayama, N. Plettenberg, M. Ziefle, and A. C. Valdez, "User preferences in recommendation algorithms: The influence of user diversity, trust, and product category on privacy perceptions in recommender algorithms," in *Proc. 12th ACM Conf. Recommender Syst.*, Vancouver, BC, Canada, 2018, pp. 306–310.

[18] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, arXiv:1411.1784. [Online]. Available: http://arxiv.org/abs/1411.1784

[19] I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, Quebec, Canada, 2014, pp. 2672–2680.

[20] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 2226–2234.

[21] Y. Xu et al., "Neural serendipity recommendation: Exploring the balance between accuracy and novelty with sparse explicit feedback," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 4, pp. 50:1–50:25, 2020.

[22] Y. Koren and R. M. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, Berlin, Germany: Springer, 2015, pp. 77–118.

[23] N. Koenigstein, P. Ram, and Y. Shavitt, "Efficient retrieval of recommendations in a matrix factorization framework," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Maui, HI, USA, 2012, pp. 535–544.

[24] B. Askari, J. Szlichta, and A. Salehi-Abari, "Variational autoencoders for top-k recommendation with implicit feedback," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Canada, 2021, pp. 2061–2065.

[25] M. A. Alshehri and X. Zhang, "Generative adversarial zero-shot learning for cold-start news recommendation," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, Atlanta, GA, USA, 2022, pp. 26–36.

[26] X. Xu et al., "Alleviating cold-start problem in CTR prediction with a variational embedding learning framework," in *Proc. ACM Web Conf.*, Lyon, France, 2022, pp. 27–35.

[27] X. Chen, C. Du, X. He, and J. Wang, "JIT2R: A joint framework for item tagging and tag-based recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, China, 2020, pp. 1681–1684.

[28] X. Huang, J. Sang, J. Yu, and C. Xu, "Learning to learn a cold-start sequential recommender," *ACM Trans. Inf. Syst.*, vol. 40, no. 2, pp. 30:1–30:25, 2022.

[29] H. Pang, F. Giunchiglia, X. Li, R. Guan, and X. Feng, "PNMTA: A pretrained network modulation and task adaptation approach for user cold-start recommendation," in *Proc. ACM Web Conf.*, Lyon, France, 2022, pp. 348–359.
[30] Y. Zhu et al., "Transfer-meta framework for cross-domain recommendation to cold-start users," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Canada, 2021, pp. 1813–1817.
[31] J. Zheng, Q. Ma, H. Gu, and Z. Zheng, "Multi-view denoising graph auto-encoders on heterogeneous information networks for cold-start recommendation," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, Singapore, 2021, pp. 2338–2348.
[32] R. Raziperchikolaei, G. Liang, and Y. Chung, "Shared neural item representations for completely cold start problem," in *Proc. 15th ACM Conf. Recommender Syst.*, Amsterdam, The Netherlands, 2021, pp. 422–431.
[33] X. Sun, T. Shi, X. Gao, Y. Kang, and G. Chen, "FORM: Follow the online regularized meta-leader for cold-start recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Canada, 2021, pp. 1177–1186.
[34] W. Wang, "Learning to recommend from sparse data via generative user feedback," in *Proc. 35th AAAI Conf. Artif. Intell., 33rd Conf. Innov. Appl. Artif. Intell., 11th Symp. Educ. Adv. Artif. Intell.*, AAAI Press, 2021, pp. 4436–4444.
[35] P. Liu, K. Shivaram, A. Culotta, M. A. Shapiro, and M. Bilgic, "The interaction between political typology and filter bubbles in news recommendation algorithms," in *Proc. Web Conf.*, Ljubljana, Slovenia, 2021, pp. 3791–3801.
[36] F. Masrour, T. Wilson, H. Yan, P. Tan, and A. Esfahanian, "Bursting the filter bubble: Fairness-aware network link prediction," in *Proc. 34th AAAI Conf. Artif. Intell., 32nd Innov. Appl. Artif. Intell. Conf., 10th AAAI Symp. Educ. Adv. Artif. Intell.*, New York, NY, USA, AAAI Press, 2020, pp. 841–848.
[37] U. Chitra and C. Musco, "Analyzing the impact of filter bubbles on social network polarization," in *Proc. 13th ACM Int. Conf. Web Search Data Mining*, Houston, TX, USA, 2020, pp. 115–123.
[38] A. Tommasel, J. M. Rodriguez, and D. Godoy, "I want to break free! recommending friends from outside the echo chamber," in *Proc. 15th ACM Conf. Recommender Syst.*, Amsterdam, The Netherlands, 2021, pp. 23–33.

Yuanbo Xu received the PhD degree in computer science and technology from Jilin University, Changchun, in 2019. He is currently an associate professor with the Department of Computer Science and Technology, Jilin University, Changchun. He is also a visiting scholar with the Management Science and Information Systems Department, Rutgers, the State University of New Jersey. His research interests include applications of data mining, recommender systems, and mobile computing. He has published some research results in journals such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Neural Networks and Learning Systems*, *ACM Transactions on Knowledge Discovery from Data* and conferences such as INFOCOM and ICDM.

En Wang received the BE degree in software engineering from Jilin University, Changchun, in 2011, the ME degree in computer science and technology from Jilin University, Changchun, in 2013, and the PhD degree in computer science and technology from Jilin University, Changchun, in 2016. He is currently an associate professor with the Department of Computer Science and Technology, Jilin University, Changchun. He is also a visiting scholar with the Department of Computer and Information Sciences, Temple University in Philadelphia. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices, and mobile crowdsensing.

Yongjian Yang received the BE degree in automatization from the Jilin University of Technology, Changchun, Jilin, China, in 1983, and the ME degree in computer communication from the Beijing University of Post and Telecommunications, Beijing, China, in 1991, and the PhD degree in software and theory of computer from Jilin University, Changchun, Jilin, China, in 2005. He is currently a professor and the PhD supervisor with Jilin University, director of Key lab under the Ministry of Information Industry, Standing director of Communication Academy, and member of the Computer Science Academy of Jilin Province. His research interests include: Theory and software technology of network intelligence management, Key technology research of wireless mobile communication and services. He participated 3 projects of NSFC, 863 and funded by the National Education Ministry for Doctoral Base Foundation. He has authored 12 projects of NSFC, key projects of Ministry of Information Industry, Middle and Young Science and Technology Developing Funds, Jilin provincial programs, ShenZhen, ZhuHai, and Changchun.

Hui Xiong (Fellow, IEEE) received the PhD degree in computer science from the University of Minnesota. He is currently a chair professor with the Thrust of Artificial Intelligence, The Hong Kong University of Science and Technology (Guangzhou). His research interests include data mining, mobile computing, and their applications in business. He has served regularly on the organization and program committees of numerous conferences, including as a Program CoChair of the Industrial and Government Track for the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), a program co-chair for the IEEE 2013 International Conference on Data Mining (ICDM), a general co-chair for the 2015 IEEE International Conference on Data Mining (ICDM), and a program co-chair of the Research Track for the 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. He received the 2021 AAAI Best Paper Award and the 2011 IEEE ICDM Best Research Paper award. For his outstanding contributions to data mining and mobile computing, he was elected an AAAS Fellow.