



# Efficient incorporation of new interactions in graph recommenders via folding-in

Viacheslav Yusupov<sup>1</sup> · Nikita Sukhorukov<sup>2,3</sup> · Evgeny Frolov<sup>1,2</sup>

Received: 12 June 2025 / Accepted in revised form: 13 November 2025

© The Author(s), under exclusive licence to Springer Nature B.V. 2025, modified publication 2026

## Abstract

Graph-based recommender systems have emerged as a powerful paradigm for personalized recommendations. However, their reliance on full model retraining to incorporate new users or new interactions creates scalability barriers. The task becomes infeasible in real-life recommender systems due to excessive time and resource costs involved. To address this limitation, we propose a fast and efficient method for updating graph-based recommender models *without full model retraining* on new data. Instead of changing all weights, we modify only small share of user representations who have new interactions. Our approach achieves a remarkable speedup of 700x over conventional model retraining approaches, drastically reducing computational overhead while maintaining the accuracy of the recommendations. Furthermore, we integrate our method into a multi-representation architecture that combines graph- and sequential-based methods to capture different user and item representations. Extensive experiments on diverse datasets demonstrate that our approach achieves state-of-the-art recommendation accuracy while maintaining the efficiency of incremental updates, outperforming existing methods in both speed and quality.

**Keywords** RecSys · Multi-representation · Folding-in · Graph neural networks.

---

✉ Viacheslav Yusupov  
vyusupov@hse.ru

Nikita Sukhorukov  
sukhorukov@airi.net

Evgeny Frolov  
frolov@airi.net

<sup>1</sup> HSE University, Moscow, Russian Federation

<sup>2</sup> AIRI, Moscow, Russian Federation

<sup>3</sup> Skoltech, Moscow, Russian Federation

## 1 Introduction

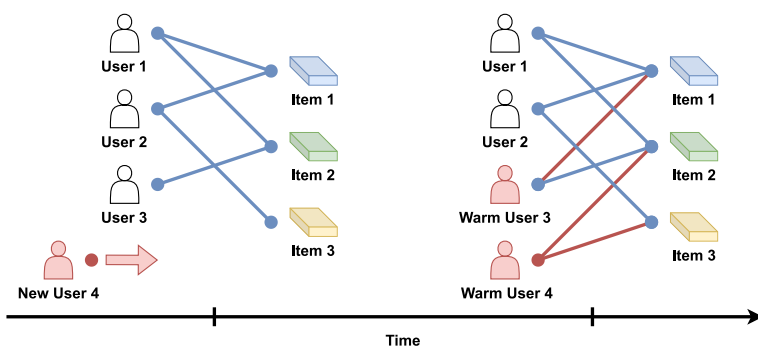
Recent advances in recommender systems have introduced two dominant neural network-based approaches: attention-based models (Kang and McAuley 2018; Sun et al. 2019) and graph neural networks (GNNs) (He et al. 2020; Mao et al. 2021). Each approach captures different aspects of user behavior.

Attention-based methods excel at capturing sequential patterns at the local user level, effectively modeling short-term user preferences. Additionally, they are fast and scalable, as they do not rely on memory-intensive user embeddings, enabling rapid updates when new interactions occur. However, their reliance on temporal patterns limits their applicability in domains where interaction sequences lack clear order, reducing their effectiveness in such cases (Klenitskiy et al. 2024).

In contrast, GNNs (Kipf et al. 2016) exploit the relational structure of user-item interactions to uncover complex higher-order dependencies, which can significantly enhance the recommendation quality. However, GNNs generally demand greater computational resources than attention-based methods, particularly when modeling large-scale interaction graphs.

Each approach has its strengths and limitations, leading researchers to propose hybrid frameworks that integrate both paradigms to leverage their advantages while mitigating their weaknesses. These unified multi-representational models have demonstrated state-of-the-art recommendation quality (Baikalov et al. 2024; Wang et al. 2022).

However, GNN-based models face a major limitation: they struggle to efficiently incorporate *warm users*, who have recently engaged in new interactions. *Warm users* refer to (1) users seen during training who have since generated new interactions, and (2) newly introduced users with some existing interaction history. Supporting such users typically requires full model retraining. The emergence of new interactions and warm users is illustrated in Fig. 1. In real-world large-scale recommender systems, where user preferences rapidly evolve and new interactions continuously arrive, full retraining is computationally prohibitive. This bottleneck significantly limits the model's adaptability, resulting in stale user representations and degrading the quality



**Fig. 1** Evolution of the interaction graph over time. Red edges indicate new interactions from warm users: existing users continuing to interact (User 3) and newly introduced users (User 4)

of recommendations over time (Sun 2023). Moreover, multi-representational models that incorporate GNNs (Baikalov et al. 2024; Wang et al. 2022) inherit this limitation, reducing their practicality in dynamic production environments.

To address the aforementioned challenges, we propose a “Folding-In” approach that enables efficient updates of user embeddings in graph-based recommender systems without requiring full model retraining. Our method enables the system to dynamically adjust user representations in response to new interactions or new users with partial histories, maintaining high recommendation quality with minimal computational cost. To demonstrate the effectiveness of our approach, we integrate it into the state-of-the-art multi-representation model, MRGSRec (Baikalov et al. 2024), enhancing its graph-based component with fast and accurate updates. Our experiments show that the proposed Folding-In mechanism achieves comparable recommendation quality compared to full model retraining while drastically reducing computational costs, making it well-suited for large-scale, dynamic environments.

Our main contributions are:

- we present a novel scheme for updating graph embeddings with new interactions, enabling efficient adaptation for both existing and newly added users without full model retraining,
- we rigorously evaluate the multi-representation recommender model using a robust data-splitting strategy,
- we perform extensive computational complexity analysis, highlighting the trade-off between recommendation quality and efficiency.

The rest of the paper is organized as follows: we discuss the problem statement and describe the “Folding-In” approach in Sect. 3. In Sect. 5, we outline the baselines and setup of the experiment. Section 6 discusses the results and findings. Related work is reviewed in Sect. 2, and Sect. 7 summarizes the paper. The code for reproducing our results is available <sup>1</sup>

## 2 Related work

In this section we conduct an overview commonly used RecSys model types: sequential-based, graph-based and multi-representational approaches, and Folding-In methods to incorporate new interactions.

### Sequential-based recommendations

The main goal of sequential recommender systems is to model the temporal dynamics of user behavior beyond static interaction patterns. User history patterns are commonly modeled by recurrent neural networks (RNNs) (Sutskever et al. 2011) and transformer (Vaswani 2017) architectures, widely used in the NLP domain. Notable examples include GRU4Rec (Hidasi 2015) and GRU4Rec+ (Hidasi and Karatzoglou 2018), which use RNNs along with modified loss functions to better model sequential

<sup>1</sup> <https://github.com/YusupovV-Lab/Folding-In-MRGSRec>

behavior. With the rise of attention mechanisms, models like BERT4Rec (Sun et al. 2019), based on the BERT architecture (Devlin et al. 2019), and DuoRec (Qiu et al. 2022), which integrates Transformers with contrastive learning techniques (Jing et al. 2023), have emerged. SASRec (Kang and McAuley 2018) and its variants that incorporate full cross-entropy loss (Klenitskiy and Vasilev 2023; Mezentsev et al. 2024; Gusak et al. 2024) have achieved state-of-the-art results in several domains. In our work, we utilize the SASRec model trained with full cross-entropy loss, a method known for its strong performance and efficiency across various recommendation benchmarks.

## Graph-based recommendations

User-item interactions can naturally be represented as a graph, with users and items as nodes, and interactions as edges. Graph-based recommendation models employ graph convolutional networks (GCNs) (Kipf et al. 2016) to learn latent representations. For instance, NGCF (Wang et al. 2019) utilizes the adjacency matrix and trainable weights, while LightGCN (He et al. 2020) simplifies it to the multiplication of adjacency matrices. The UltraGCN model (Mao et al. 2021) is an even more simplified version of LightGCN, where the adjacency matrix is distilled into the diagonal one. Additionally, GATRec (Wu et al. 2019) utilizes graph attention networks (GATs) (Veličković et al. 2018) to capture more nuanced relational information.

Recent studies have also explored enhancing graph-based recommenders with self-supervised learning. For example, studies showed that self-supervision can boost LightGCN's performance (Wu et al. 2021), while SGCL (Zhang et al. 2025) aligns supervised recommendation loss with contrastive self-supervised loss. Other approaches integrate multiple feedback and entity types into unified architectures, such as SDGL (Li et al. 2025).

In this work, we employ the LightGCN model—a lightweight and effective graph-based recommendation approach known for its strong performance across diverse datasets. Despite the effectiveness of graph-based approaches, they struggle to effectively capture new interactions among existing users as well as interactions involving new users.

## Multi-representational recommendations

To combine the strengths of sequential- and graph-based methods, multi-representational models integrate both approaches. Examples include MRGSRec (Baikalov et al. 2024), MCLSR (Wang et al. 2022), and STAM (Yang et al. 2022), which leverage both components in a unified framework. The graph component captures indirect user-item relationships and global structural patterns in the dataset, while the sequential component focuses on user-specific interaction histories to predict future behavior. While the sequential part excels at modeling personal behavior, it lacks a view of the overall dataset structure. Combining both perspectives, personal and global, leads to significant improvements in recommendation quality (Baikalov et al. 2024). Despite the fact that MRGSRec is the effective approach, it suffer from

inability to capture new interactions. In this work, we develop the MRGSRec method with Folding-In approach.

## New interactions

Given the high frequency of user interactions in real-world systems, it is common for recommendation models to encounter either entirely new users who begin interacting with items after training, or existing users who generate additional interactions that were not included in the training set. To handle such cases, existing approaches (He et al. 2016; Chen et al. 2020) typically employ *Folding-In* techniques (Bellogin et al. 2014; Xin et al. 2017; Hoecker and Kartvelishvili 1996) or reinforcement learning (RL)-based methods (Yang et al. 2020), which update user and item representations incrementally as new interactions arrive. However, to the best of our knowledge, no previous work has addressed efficient user representation updates for *warm-start* scenarios (Gogna and Majumdar 2015) in graph-based recommendation models.

While much prior work focuses on designing new models to improve evaluation metrics, we propose a practical and generalizable method for updating recommendations *without requiring full model retraining*. Instead of re-optimizing all model parameters, our approach updates only a small subset of user embeddings, yielding substantial efficiency gains while preserving recommendation quality. The method is broadly applicable to recommender systems that incorporate a graph component, though we focus on *multi-representational models*, which are more resilient to challenges related to updating recommendations based on graph information. Additionally, in this work, we evaluate all models using a carefully constructed dataset split that *prevents future information leakage*, ensuring that the evaluation more accurately reflects real-world scenarios. While we evaluated our approach only in the recommender system domain, our approach is universal and could be applied to new entities on knowledge graphs (Peng et al. 2023) or new users on social networks (Liang et al. 2018).

## 3 Preliminaries

Graph-based approaches effectively capture high-order patterns in user behavior. However, they often face challenges in quickly adapting to rapid shifts in user interests, which can occur frequently. This paper introduces an efficient method for updating user embeddings, enhancing the adaptability of graph-based recommender models. Our approach is particularly advantageous for multi-representational models, enabling them to achieve state-of-the-art performance while significantly reducing inference time in comparison to full model retraining. This improvement makes these models more practical and scalable for real-world applications.

In the following sections, we provide an overview of the sequential and graph components of the MRGSRec (Baikalov et al. 2024) framework, which forms the basis of our research. These components are essential to accurately capture user preferences and enhancing personalized experiences across various domains.

### 3.1 Graph-based component

Graph-based recommendation systems utilize the structural properties of graph data to improve model performance. In this approach, user-item interactions are represented as a bipartite graph, with one set of nodes representing users and the other representing items. Edges between users and items denote interactions, enabling the model to capture complex relational information that extends beyond mere co-occurrence patterns.

Similarly to most of the existing recommender system models, users and items are presented as embedding vectors. Here and further, the user  $u$  embeddings will be indexed by  $\mathbf{e}_u \in \mathbb{R}^d$ , and item  $i$  one by  $\mathbf{e}_i \in \mathbb{R}^d$ , where  $d$  is the embedding dimension. Therefore, the embedding matrix of the graph part is formed as follows:

$$\mathbf{E} = [\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \dots, \mathbf{e}_{u_M}, \mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \dots, \mathbf{e}_{i_N}] \in \mathbb{R}^{(N+M) \times d},$$

where  $M$  is the number of users and  $N$  is the number of items.

To incorporate the graph structure, we employ LightGCN (He et al. 2020) as the graph encoder. Let us define the graph adjacency matrix  $\mathbf{A} \in \mathbb{R}^{(N+M) \times (N+M)}$  of the bipartite user-item graph. To construct  $\mathbf{A}$  we first introduce a binary matrix  $\mathbf{R} \in \mathbb{R}^{N \times M}$  with elements  $r_{ij}$  equal to 1 if the user  $i$  has interaction with the item  $j$  and 0 otherwise. The adjacency matrix  $\mathbf{A}$ , which contains the global information about all user-item interactions, is a block matrix defined as follows:

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(N+M) \times (N+M)}. \quad (1)$$

To introduce the  $k$ -th graph encoder layer, we first define its input as  $\mathbf{E}^{(k-1)}$ , where  $\mathbf{E}^{(0)}$  is the embedding matrix  $\mathbf{E}$ . Therefore the output  $\mathbf{E}^{(k)}$  of graph encoder layer is recursively defined as:

$$\mathbf{E}^{(k)} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{E}^{(k-1)} = \hat{\mathbf{A}} \mathbf{E}^{(k-1)}, \quad (2)$$

where  $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$  is the symmetrically normalized matrix. In this formula  $\mathbf{D}$  is a diagonal matrix where  $D_{ii} = \sum_{j=1}^{N+M} A_{ij}$ . The final embedding matrix of the graph part with  $k$  layers is written as:

$$\mathbf{E}^{(F)} = \mathbf{E}^{(0)} + \mathbf{E}^{(1)} + \dots + \mathbf{E}^{(k)},$$

Which could be rewritten using (2) as follows:

$$\mathbf{E}^{(F)} = \mathbf{E}^{(0)} + \hat{\mathbf{A}} \mathbf{E}^{(0)} + \hat{\mathbf{A}}^2 \mathbf{E}^{(0)} + \dots + \hat{\mathbf{A}}^k \mathbf{E}^{(0)} = (\mathbf{I} + \hat{\mathbf{A}} + \hat{\mathbf{A}}^2 + \dots + \hat{\mathbf{A}}^k) \mathbf{E}^{(0)}. \quad (3)$$

For brevity, we introduce  $\hat{\mathbf{L}}^{(k)} = (\mathbf{I} + \hat{\mathbf{A}} + \hat{\mathbf{A}}^2 + \dots + \hat{\mathbf{A}}^k)$ . So the equation (3) could be rewritten as:

$$\mathbf{E}^{(F)} = \hat{\mathbf{L}}^{(k)} \mathbf{E}^{(0)}. \quad (4)$$

As seen in (3), the graph part of MRGSRec approach contains only  $(N + M) \times d$  trainable parameters in Embedding layers, while all other graph encoder layers are parameter-free.

Using the graph encoder defined in Equation (4), predictions are made by computing a score between user  $u$  and item  $i$ :

$$y_{ui} = \mathbf{e}_u^\top \mathbf{e}_i, \quad (5)$$

where  $\mathbf{e}_u$  is a user embedding and  $\mathbf{e}_i$  is an item embedding.

LightGCN (He et al. 2020) and graph part of MRGSRec (Baikalov et al. 2024) are trained with *BPR* (Rendle et al. 2009) loss:

$$\mathcal{L}_G = \mathcal{L}_{\text{BPR}} + \lambda \|\mathbf{E}^{(0)}\| = - \sum_{u=1}^M \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln(\sigma(y_{ui} - y_{uj})) + \lambda \|\mathbf{E}^{(0)}\|, \quad (6)$$

where  $\mathcal{N}_u$  is the set of items user  $u$  interacted with and  $\lambda$  is a hyper-parameter. The scores  $y_{ui}$ ,  $y_{uj}$  are calculated via (5) and indicate the relevance of the items  $i$ ,  $j$  to the user  $u$ .

### 3.2 Sequential-based component

Recent studies highlight the effectiveness of sequential recommender systems, which leverage the ordered interaction sequences to capture users' evolving interests. The following subsections describe the layers of sequential component of MRGSRec (Baikalov et al. 2024) framework based on the SASRec (Kang and McAuley 2018) model.

For user  $u$  the interaction sequence  $S_u$  is defined as  $S_u = (s_1, s_2, \dots, s_n)$ , where the sequence length  $n$  is fixed for each user. If the user has fewer than  $n$  interactions, the sequence is padded with a special token to the length  $n$ . In case the user has more than  $n$  interactions, only the most recent  $n$  interactions are kept.

We define the item embedding matrix  $\mathbf{E}^I \in \mathbb{R}^{N \times d}$ , where  $N$  is the number of unique items and  $d$  is the dimension of the embedding vectors. The corresponding embedding matrix for sequence  $S$  is matrix  $\mathbf{I}^S \in \mathbb{R}^{n \times d}$  for sequence  $S$ , where each row is given by  $\mathbf{I}_i^S = \mathbf{E}_{S_i}^I$ , and  $S_i$  denotes the  $i$ -th element of sequence  $S$ .

To incorporate positional information into the model, we add learnable positional embeddings  $\mathbf{P} \in \mathbb{R}^{n \times d}$  to item embeddings  $\mathbf{I}^S$ . Therefore the output of embedding layer for the sequence  $S$  is defined as follows:

$$\hat{\mathbf{E}}^S = \mathbf{I}^S + \mathbf{P} = \begin{bmatrix} \mathbf{I}_1^S + \mathbf{P}_1 \\ \mathbf{I}_2^S + \mathbf{P}_2 \\ \dots \\ \mathbf{I}_n^S + \mathbf{P}_n \end{bmatrix}.$$

The sequential user embedding  $\mathbf{e}_u = \text{SASRec}(\hat{\mathbf{E}}^{S_u})$  is obtained by processing the user's interaction history  $S_u$  through the SASRec model, where  $\hat{\mathbf{E}}^{S_u}$  encodes the sequence of items interacted with by user  $u$ .

### 3.2.1 Self-attention block

The following subsection describes the *Self-Attention Block* (Vaswani 2017), a mechanism initially developed for sequential data processing in NLP tasks and subsequently adapted for recommendation systems. Its effectiveness arises from the structural similarity between natural language and user interaction sequences, where the interpretation of the current element depends on its preceding context. This architectural component captures these contextual dependencies by dynamically assigning weights to relevant previous elements when processing each position in the sequence.

First we define scaled dot-product attention as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \quad (7)$$

where  $\mathbf{Q}$ ,  $\mathbf{K}$  and  $\mathbf{V}$  represent queries, keys and values, respectively. This mechanism calculates the weighted sum of value vectors with weights equal to:

$$\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right),$$

where the scale factor  $\sqrt{d}$  is used to avoid large values in the  $\mathbf{Q}\mathbf{K}^\top$  matrix.

The first layer of *Self-Attention Block* is *Self-Attention layer* or *SA* for shorts defined as:

$$\mathbf{S} = \text{SA}(\hat{\mathbf{E}}) = \text{Attention}(\hat{\mathbf{E}}\mathbf{W}^Q, \hat{\mathbf{E}}\mathbf{W}^K, \hat{\mathbf{E}}\mathbf{W}^V),$$

where  $\mathbf{W}^Q$ ,  $\mathbf{W}^K$ ,  $\mathbf{W}^V \in \mathbb{R}^{d \times d}$  are projection matrices.

The second layer of *Self-Attention Block* is *Point-Wise Feed-Forward Network* or *FFN* for shorts, which adds nonlinearity to the model. The output of the *FFN* layer applied to the  $i$ -th row of *SA* equals  $F_i$ :

$$F_i = \text{FFN}(S_i) = \text{ReLU}(S_i \mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)},$$

where  $\mathbf{W}^{(1)}$ ,  $\mathbf{W}^{(2)} \in \mathbb{R}^{d \times d}$  are trainable weight matrices and  $\mathbf{b}^{(1)}$ ,  $\mathbf{b}^{(2)} \in \mathbb{R}^d$  are trainable bias vectors.

To avoid the model overfitting, the *Dropout* (Srivastava et al. 2014) layers are used after *SA* and *FFN*. This layer makes all trainable weights equal to 0 with probability  $p$  and leaves them without changes with probability  $1 - p$ .

To normalize the inputs of the layers *SA* and *FNN* the *LayerNorm* (Ba 2016) is used. The operation for *LayerNorm* applied to the input  $x$  is defined as:

$$\text{LayerNorm}(x) = \alpha \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta,$$

where  $\mu$  and  $\sigma$  are the mean and variance of the input  $x$ ,  $\alpha$  is the learnable scaling factor,  $\beta$  is a trainable bias vector and  $\epsilon$  a small positive number to increase the numerical stability of the operation. To avoid vanishing gradients, the residual connections are used around the *SA* and *FFN* layers.

With mentioned improvements the *Self-Attention layer* and *Feed-Forward Network* of  $k - th$  *Self-Attention block* could be rewritten as follows:

$$\mathbf{S}^{(k)} = \text{SA}(\mathbf{F}^{(k-1)}) = \mathbf{F}^{(k-1)} + \text{Dropout}(\text{SA}(\text{LayerNorm}(\mathbf{F}^{(k-1)}))),$$

and

$$\mathbf{F}_i^{(k)} = \text{FFN}(\mathbf{S}_i^{(k)}) = \mathbf{S}_i^{(k)} + \text{Dropout}(\text{FFN}(\text{LayerNorm}(\mathbf{S}_i^{(k)}))), \quad (8)$$

where  $\mathbf{F}^{(0)} = \hat{\mathbf{E}}$  and  $i \in (1, 2, \dots, n)$ .

The relevance score between user  $u$  and item  $i$  is calculated accordingly:

$$r_{ii}^{(u)} = \mathbf{F}_i^{(u)} \mathbf{e}_i, \quad (9)$$

where  $\mathbf{F}_i^{(u)}$  is the output of sequential encoder (8) for sequence  $s_i$  of user  $u$  interactions and  $\mathbf{e}_i$  is an item  $i$  embedding

The SASRec (Kang and McAuley 2018) model was trained with the *BCE* loss, however, the recent works (Gusak et al. 2024; Mezentsev et al. 2024) demonstrated that utilization of *Cross-Entropy* (CE) loss significantly improves the recommendation metrics. The same CE loss was utilized to train the sequential part of the MRGSRec model. In the following formulas we demonstrate *BCE* loss:

$$\mathcal{L}_{BCE} = - \sum_{u=1}^M \sum_{i \in \mathcal{N}_u} \ln(\sigma(r_{ii}^{(u)})) + \ln(1 - \sigma(r_{i-}^{(u)})) \quad (10)$$

and *CE* loss

$$\mathcal{L}_{CE} = - \sum_{u=1}^M \sum_{i \in \mathcal{N}_u} \ln \frac{\exp(r_{ii}^{(u)})}{\sum_{j \in \mathcal{I}} \exp(r_{ij}^{(u)})}. \quad (11)$$

In these formulas,  $\mathcal{N}_u$  is a set of items user  $u$  has interacted with, and  $\mathcal{I}$  is a set of all items. Scores  $r_{ii}^{(u)}$ ,  $r_{ij}^{(u)}$  and  $r_{i-}^{(u)}$  are calculated using (9), and  $r_{i-}^{(u)}$  denotes the score between user  $u$  and random negative item for this user.

### 3.3 Multi-representational approach

To capture information from the global and personal dataset, the Multi-Representational Graph-based and Sequential-based Recommender model combines Graph 3.1 and Sequential 3.2 via the Fusion layer 3.3.1. This hybrid approach synergies global graph-structured information with personalized sequential patterns, mirroring multi-modal systems that effectively combine diverse representations.

The MRGSRec framework could be used with different encoders with following signature:

$$e_F^u || E_F^u = \text{AnyEncoder}(e^u || E^u), \quad (12)$$

where  $e^u$  and  $e_F^u$  initial and final local embeddings of user  $u$ ,  $E^u$  and  $E_F^u$  initial and final global embeddings and  $|| : (\mathbb{R}^n, \mathbb{R}^m) \rightarrow \mathbb{R}^{n+m}$  is concatenation operator. In our work, we utilize the LightGCN (He et al. 2020) and SASRec (Kang and McAuley 2018) encoders as effective and commonly used recommendation models.

#### 3.3.1 Fusion layer

After processing the user and item embeddings, the Graph (3.1) and Sequential (3.2) encoders return embeddings  $e_u^g \in \mathbb{R}^d$  and  $e_u^s \in \mathbb{R}^d$ , respectively. The fusion layer converts the embeddings into one  $e_u^f \in \mathbb{R}^d$  embedding according to formula:

$$e_u^f = \mathbf{W}_2 \text{ReLU}(\mathbf{W}_1(e_u^g || e_u^s)),$$

where  $\mathbf{W}_1 \in \mathbb{R}^{4d \times 2d}$  and  $\mathbf{W}_2 \in \mathbb{R}^{d \times 4d}$  are trainable matrices and  $|| : (\mathbb{R}^d, \mathbb{R}^d) \rightarrow \mathbb{R}^{2d}$  is the vector concatenation function.

The relevance  $r_{ui}$  of the item  $i$  for the user  $u$  is equal to:

$$r_{ui} = (e_u^f)^\top e_i,$$

where  $e_u^f$  is a user  $u$  fusion embedding and  $e_i$  is an item  $i$  embedding.

#### 3.3.2 Loss functions

To enable end-to-end usage of graph and sequential model components, the MRGSRec approach utilizes a loss function constructed from four components.

*Graph loss*  $\mathcal{L}_G$  utilizes BPR loss function with regularization (6) similar to one, that was used in LightGCN (He et al. 2020).

*Sequential loss*  $\mathcal{L}_S$  is an cross-entropy (CE) loss (11) which improves sequential encoder 3.2 effectiveness compared to BCE (10).

*Fusion loss*  $\mathcal{L}_F$  is based on fused embeddings combine information from graph and sequential representations:

$$\mathcal{L}_F = - \sum_{u=1}^M \ln \frac{\exp(r_{ui})}{\exp(r_{ui}) + \sum_{j \in \mathcal{N}_u^-} \exp(r_{uj})},$$

where  $\mathcal{N}_u^-$  is the set of negative sampled items for the user  $u$ . The scores  $r_{ui}$  and  $r_{uj}$  are obtained using the formula (13).

*Contrastive loss*  $\mathcal{L}_C$  assists the model in capturing the difference between graph and sequential representations of items. The *Contrastive loss* is calculated as follows:

$$\mathcal{L}_C = - \sum_{u=1}^M \sum_{i \in \mathcal{N}_u} \ln \frac{\exp((\mathbf{e}_i^s)^\top \mathbf{e}_i^g)}{\sum_{j \in \mathcal{N}_u} \exp((\mathbf{e}_i^s)^\top \mathbf{e}_j^g)},$$

where  $\mathbf{e}_i^s$  and  $\mathbf{e}_i^g$  represent the sequential- and graph-based embeddings of item  $i$ , respectively. This loss function is designed to differentiate between these two representations while minimizing interference of information between the different approaches.

The overall loss function is given as follows:

$$\mathcal{L}_{\text{MRGSRec}} = \alpha \mathcal{L}_S + \beta \mathcal{L}_G + \gamma \mathcal{L}_F + \delta \mathcal{L}_C, \quad (13)$$

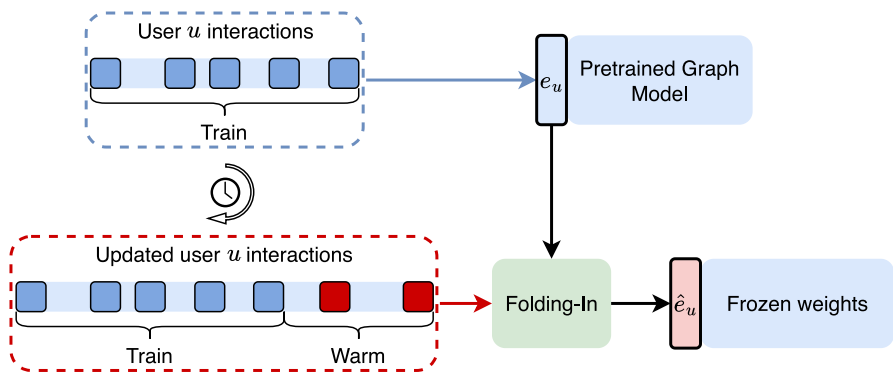
where  $\alpha, \beta, \gamma, \delta$  are MRGSRec hyper-parameters. The model is trained with *Adam* (Kingma and Ba 2014) optimizer.

## 4 Proposed approach

Now we are ready to describe our *Folding-In* procedure in detail (see Fig. 2). To update user  $u$  embedding  $\mathbf{e}_u$  we freeze the embeddings for all other items and users and minimize the loss function:

$$\mathcal{L}(\mathbf{e}_u) = \mathcal{L}_{\text{BPR}}(\mathbf{e}_u) + \lambda \|\mathbf{E}_u - \mathbf{e}_u\|_2^2 \quad (14)$$

with  $n \in \{1, 2, \dots, 5\}$  steps of *Adam* (Kingma and Ba 2014; Amari 1993) optimization method, where  $n$  is a hyperparameter. In the loss function  $\mathcal{L}$  given in Equation (14),  $\mathcal{L}_{\text{BPR}}$  is a *BPR* (6) loss associated with the models LightGCN (He et al. 2020). The vector  $\mathbf{E}_u$  denotes the initial user  $u$  embedding with noise, and  $\lambda$  is the hyperparameter controlling the regularization term. The batch for minimization  $\mathcal{L}$  comprises both positive interactions of the new user  $u$  ( $n_u$  interactions) and randomly sampled positive interactions from previous training data ( $n_r$  interactions). To generate negative samples for *BPR*, we sampled  $n_u$  negative interactions for user  $u$  and  $n_r$  negative user-item pairs are selected randomly.



**Fig. 2** Overview of the folding-in process applied to a pretrained graph model. Only the embedding  $e_u$  for user  $u$  is updated based on new warm interactions (highlighted in red). All other user embeddings and the model weights remain unchanged

To prevent overfitting to the newest user interactions, we firstly set  $E_u$  embedding as follows:

$$E_u = (1 - \mu)E_u^{(0)} + \mu \frac{1}{M} \sum_{i=1}^M E_{u_i}, \tag{15}$$

where  $E_u$  is an initial embedding of user  $u$ ,  $\mu$  is hyperparameter, that controls how closely  $E_u$  remains to its initial embedding  $E_u^{(0)}$ . Due to the high risk of overfitting, especially with a large number of steps, we restrict the regularization parameters to the ranges  $\lambda \in [0.1, 10]$  and  $\mu \in [0.001, 0.1]$ .

Similar strategies can be applied to other models, such as various graph-based recommender systems (Mao et al. 2021; Wang et al. 2019) or multi-representational models (Baikalov et al. 2024; Wang et al. 2022). In these cases, the *BPR* loss is replaced with the specific loss function of the respective model.

To apply the mentioned Folding-In procedure to MRGSRec model we replace in equation (14) the  $\mathcal{L}_{BPR}$  with MRGSRec composite loss  $\mathcal{L}_{MRGSRec}$  (13). Therefore the optimization task is formulated as follows:

$$\mathcal{L}(e_u) = \mathcal{L}_{MRGSRec}(e_u) + \lambda \|E_u - e_u\|_2^2, \tag{16}$$

where  $e_u$  is an embedding of user  $u$  and  $E_u$  is defined following the formula (15) with fixed hyperparameter  $\mu$ . An overview of the complete approach is presented in Fig. 3.

To minimize  $\mathcal{L}(e_u)$  and update  $e_u$ , we make the  $n \in \{1, 2, \dots, 5\}$  steps of the *Adam* (Kingma and Ba 2014) optimizer with fixed learning rate  $lr_{FI}$  equal for all users in one data batch for each user. This batch is formed in a way similar to (14) one.

The use of *SGD* methods enables us to update the user embeddings of individuals who have new interactions after the moment of the recommender model training. This Folding-In approach enhances the model’s flexibility and universality, allowing it to seamlessly adapt to evolving user data.

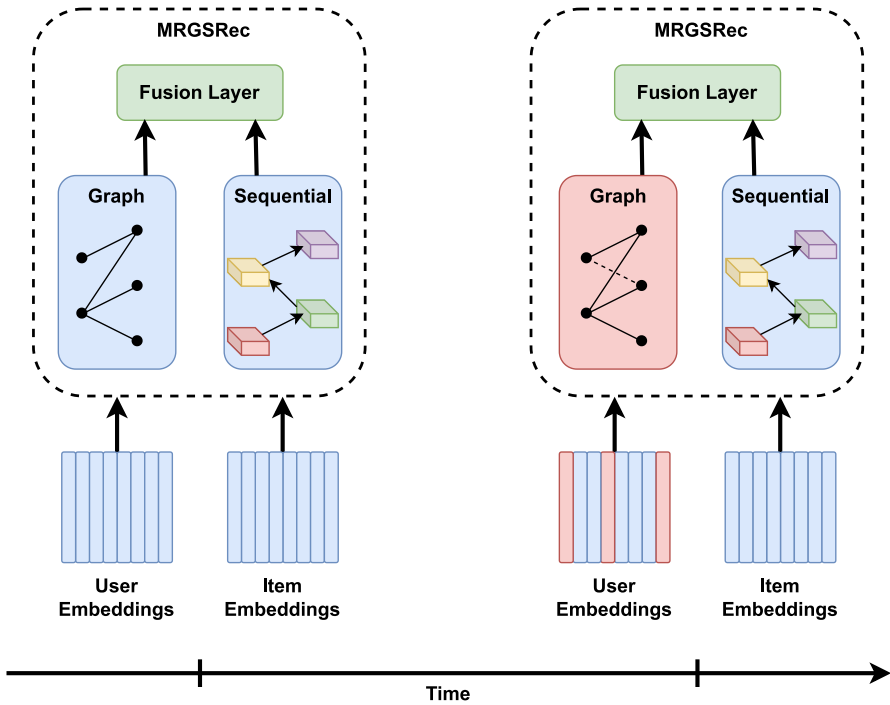


Fig. 3 MRGSRec model with the proposed graph Folding-In inference, with modified components highlighted in red. The left panel represents the model prior to the inclusion of new interactions, and the right panel shows the updated model after integrating warm user data. The sequential component is updated in a straightforward manner, as discussed in Sect. 5.2

## 5 Experiments

In this section, we outline our experimental setup.

### 5.1 Datasets

We evaluated our model on four real-world datasets: MovieLens-1M (ML-1M) (Harper and Konstan 2015), Amazon Beauty, Amazon Clothing, and Amazon Sports (He and McAuley 2016). These datasets are widely used in recommender systems research and cover various domains. The information about datasets could be found in Table 1.

Data splitting has long been a topic of debate in recommender systems research. Many academic papers (He et al. 2020; Sun et al. 2019) employ unrealistic data-splitting setups that can introduce data leaks, an issue where training data inadvertently contains information about test interactions. This leads to unfair and biased evaluations of the model performance. Figure 4 illustrates the various data-splitting methods discussed. For example, the commonly used random-split approach assigns random interactions to the test set while the remaining interactions form the training set. This

**Table 1** Dataset statistics

Dataset	# users	# items	# actions	Avg. length
ML-1 M	6,040	3,706	1,000,209	165.6
Beauty	22,363	12,101	198,502	8.9
Clothing	35,598	18,357	296,337	7.1
Sports	39,387	23,033	278,677	8.3

method introduces significant temporal inconsistencies, as training examples may include interactions that occur after the test interactions, allowing models to leverage future information is an unrealistic scenario. Such violations are particularly harmful to sequential models, which depend on the chronological order of interactions to accurately capture user behavior patterns (Kang and McAuley 2018; Sun et al. 2019; Hidasi 2015).

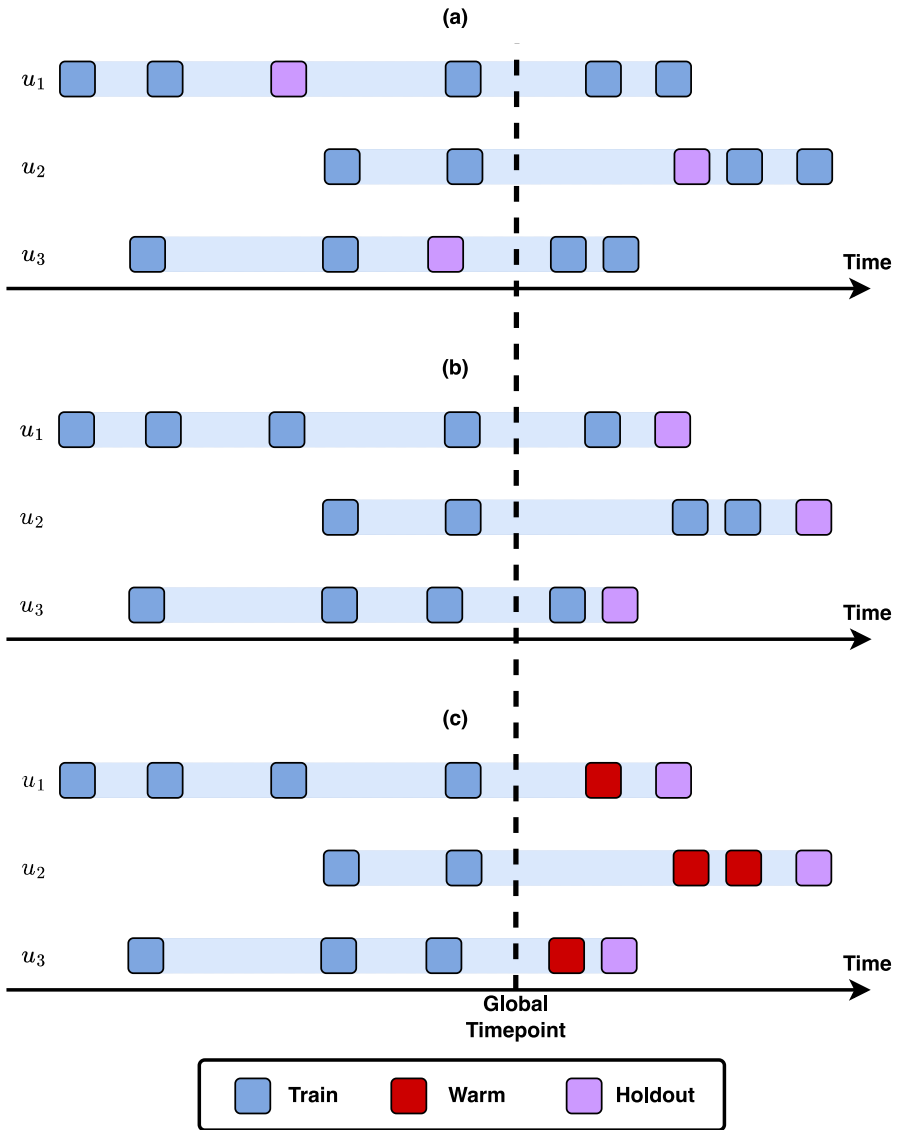
Although the Leave-Last-Out (LLO) strategy mitigates these issues by reserving each user's final interaction for testing, it fails to address cross-user data leakage. Specifically, consider two users  $u_1$  and  $u_2$  (see Fig. 4b): if  $u_2$ 's training interactions occur after  $u_1$ 's test interaction, the model may exploit  $u_2$ 's future behavior to predict  $u_1$ 's past actions.

To prevent the information leak described above, we employ a data split based on a combination of a global timepoint (GT) and the LLO split, referred to as the *GT+LLO split*. This approach closely mimics real-world recommender system training and optimization scenarios.

To create the *GT+LLO split* used in this work, we sort user-item interactions by timestamp and set the time point  $T$  based on the percentage of interactions occurring before this timestamp. All interactions occurring before  $T$  are assigned to the *train* set (see Fig. 4c). Users with interactions after  $T$  are marked as test users, and their last interaction forms the holdout set. During the training process, the model learns from the train interactions prior to  $T$ . During inference for each test user, we take its complete interaction history (both train and warm segments) and generate a top- $k$  recommendations list. This is then evaluated against the holdout interaction, which remains hidden from the model during both training and inference. The same procedure is applied to create the validation set for model hyperparameter tuning. We explicitly exclude the cold-start problem from consideration; all new interactions involve items that were already present in the training catalogue. This splitting strategy prevents information leakage while closely simulating real-world recommendation scenarios, ensuring a realistic and robust evaluation of the model performance.

## 5.2 Evaluation

First, we train our model on the *train* set of interactions (see Fig. 4c), where user and item embeddings are randomly initialized according to their IDs in the training data. To evaluate on the *holdout* set, we **independently** update the user embeddings using their new interactions labeled as *warm*, following the *Folding-In* procedure. If a user does not appear in the training data (i.e., their interactions occur only after



**Fig. 4** Comparison of different data splitting techniques on the same dataset: (a) Random splitting, (b) Leave-Last-Out (LLO) splitting, and (c) Global Timepoint + LLO splitting. Each row represents a user's interaction history over time, where individual interactions are shown as squares

the training timestamp), we randomly initialize their embedding and then apply our proposed update method. This approach enables us to update user representations without retraining the entire model on the combined *train* and *warm* data.

In our *Folding-In* process, we apply the update rule defined in Equation (16) for each test user. The model is trained on a single batch of data that includes both *train* and *warm* interactions for the user  $u$ .

For the sequential component (SASRec in our case), this process is naturally integrated, as the user embedding is derived from the sequence of the item embeddings. Since the item embeddings remain fixed during Folding-In, the user representation is automatically updated as the interaction sequence evolves.

### 5.3 Metrics

We evaluated model performance on the top- $N$  recommendation task by comparing the list of  $N$  recommended items against the ground truth item in holdout (see Fig. 4). To assess the effectiveness of the models, we used the accuracy-based Hit Rate ( $\text{HR}@N$ ) and ranking-based Normalized Discounted Cumulative Gain ( $\text{NDCG}@N$ ) (Wang et al. 2013).

**Hit Rate** ( $\text{HR}@N$ ) measures the proportion of test cases where the ground-truth item appears within the top- $N$  recommendations. Since each user in our test set has a single held-out item,  $\text{HR}@N$  is equivalent to  $\text{Recall}@N$ .

**NDCG** ( $\text{NDCG}@N$ ) is a ranking-sensitive metric that assigns greater importance to correctly predicted items appearing earlier in the recommendation list, using a logarithmic discounting factor. If the ground truth item ranks beyond position  $N$ , it contributes zero to the score.

We report both metrics for  $N \in \{5, 10\}$ , averaging the results across all users in the test set.

### 5.4 Baselines

To demonstrate the effectiveness of MRGSRec method with the *Folding-In* procedure (MRGSRec+ for shorts) on the new dataset split, we compare this method with strong graph-based and sequential-based baselines:

- **LightGCN** (He et al. 2020): Light Graph Convolution Network—the graph-based method utilizing graph encoder and graph embedding layers described in Sect. 3.1. The recommendations are made base on the score value  $r_{ui} = \mathbf{e}_u \mathbf{e}_i^\top$ , where  $\mathbf{e}_u$  and  $\mathbf{e}_i$  are user and item embeddings respectfully.
- **UltraGCN** (Mao et al. 2021): The Ultra Simplification of Graph Neural Network—the simplification of the LightGCN model (He et al. 2020), where the graph adjacency matrix is reduced to the diagonal matrix with nodes degree on the diagonal.
- **SASRec** (Kang and McAuley 2018): Self-Attentive Sequential Recommendation—the sequential-based methods with embeddings and Self-Attention blocks described in Sect. 3.2. The model is trained to minimize Binary Cross-Entropy (BCE) loss. The model score for user  $u$  with sequence of interactions  $(s_1, s_2, \dots, s_t)$  and item  $i$  is equal to  $r_{uit} = \mathbf{F}_t(\mathbf{E}_i^I)^\top$ , where  $\mathbf{F}_t$  the output of the last Self-Attention Block element  $s_t$  of the sequence, and  $\mathbf{E}_i^I$ —is the embedding of item  $i$ . To find the optimal hyperparameters for the model, we utilized a random grid search with 60 hyperparameter sets for each model.
- **SASRec+** (Klenitskiy and Vasilev 2023): Self-Attentive Sequential model similar to SASRec with full Cross-Entropy (CE) instead of Binary Cross-Entropy. The

SASRec+ is known as the *strongest current sequential baseline* (Gusak et al. 2024; Mezentsev et al. 2024).

- MRGSRec (Baikalov et al. 2024): Multi-Representation Graph- and Sequence-based approach combines Graph 3.1 and Sequential 3.2 encoders with shared embeddings. The fusion layer 3.3.1 is utilized to combine the output of the graph and the sequential encoders. For brevity, we refer to the MRGSRec model as Zero when it is trained exclusively on the initial training data and generates predictions without updating user embeddings based on new interactions from *warm users*. In contrast, the Full variant refers to MRGSRec trained on the union of the training and warm data, where model retraining is performed as part of the inference step. These two variants enable us to compare MRGSRec against our proposed method in terms of both recommendation quality and computational overhead.

## 6 Results and discussion

In the following section in Table 2, we demonstrate the comparison of our MRGSRec+ with existing baselines on newly developed datasets splits.

### 6.1 Performance comparison

We denote the multi-representational model with the Folding-In procedure as MRGSRec+. The standard MRGSRec model (without Folding-In) utilizes user embeddings trained only on the training data, without incorporating warm interactions (see Fig. 4). The Folding-In approach enables the integration of these additional interactions, allowing user embeddings to be dynamically updated.

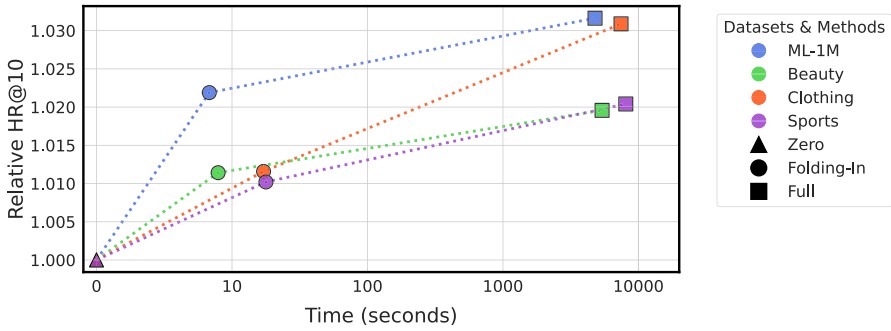
Our MRGSRec+ approach demonstrates the best quality over all baselines, significantly outperforming existing methods by effectively incorporating new user interactions in real time. The use of the *Folding-In* procedure not only enables fast adaptation to dynamic user preferences, but also improves the quality of recommendations compared to the MRGSRec method across all metrics and datasets. The results in Table 2 highlight the importance of *Folding-In* in multi-representation learning for real-world setups, where the ability to quickly integrate new interactions leads to a measurable boost in recommendation accuracy. Unlike traditional approaches, our method efficiently updates user representations without costly and time-consuming retraining, ensuring both scalability and enhanced personalization.

Additionally, we confirm the effectiveness of the MRGSRec (Baikalov et al. 2024) model under more robust dataset split, as it surpasses both sequential models (Kang and McAuley 2018) and graph-based methods (He et al. 2020).

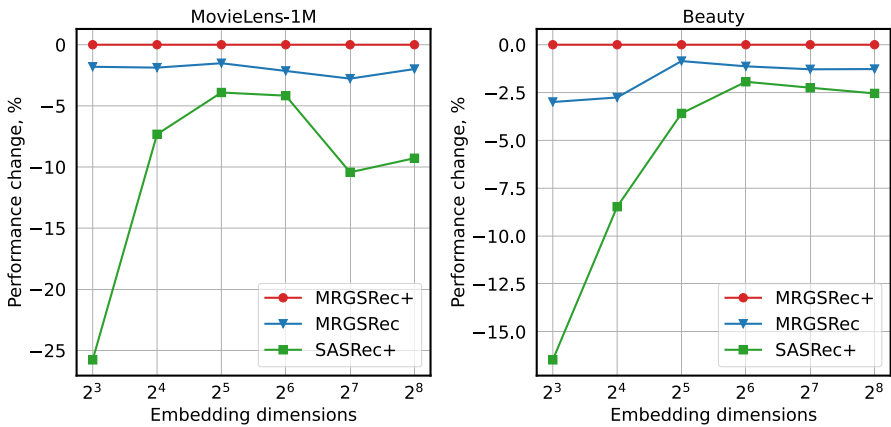
Furthermore, we compare the performance of our *Folding-In* approach in comparison with full model retraining. In this experiment, we train the MRGSRec model on the union of train and warm data. The results of the experiments are demonstrated in Fig. 2. As demonstrated, the Folding-In technique achieves superior performance metrics compared to the MRGSRec without Folding-In procedure (Zero for short)

**Table 2** Performance comparison of different recommender models across datasets. Best metric values are highlighted in **bold** and second best values are underlined. H – HR metric, N – NDCG metric

Dataset	Metric	LightGCN	UltraGCN	SASRec	SASRec+	MGRSRec	MGRSRec+	Uplift, %
Beauty	H@5	0.0227	0.0231	0.0304	0.0321	<u>0.0326</u>	<b>0.0329</b>	0.92
	H@10	0.0421	0.0426	0.0584	0.0608	<u>0.0613</u>	<b>0.0620</b>	1.14
	N@5	0.0164	0.0165	0.0218	0.0239	<u>0.0241</u>	<b>0.0244</b>	1.24
	N@10	0.0245	0.0248	0.0301	0.0320	<u>0.0324</u>	<b>0.0328</b>	1.23
Clothing	H@5	0.0074	0.0075	0.0117	0.0136	<u>0.0141</u>	<b>0.0144</b>	2.13
	H@10	0.0112	0.0115	0.0226	0.0251	<u>0.0259</u>	<b>0.0262</b>	1.15
	N@5	0.0051	0.0052	0.0078	0.0091	<u>0.0092</u>	<b>0.0094</b>	2.18
	N@10	0.0073	0.0075	0.0105	0.0122	<u>0.0125</u>	<b>0.0126</b>	0.80
Sports	H@5	0.0158	0.0160	0.0192	0.0206	<u>0.0210</u>	<b>0.0212</b>	0.95
	H@10	0.0286	0.0290	0.0354	0.0388	<u>0.0392</u>	<b>0.0396</b>	1.01
	N@5	0.0105	0.0107	0.0132	0.0137	<u>0.0139</u>	<b>0.0140</b>	0.72
	N@10	0.0147	0.0149	0.0195	0.0206	<u>0.0208</u>	<b>0.0211</b>	1.44
ML-1M	H@5	0.0340	0.0348	0.0402	0.0415	<u>0.0423</u>	<b>0.0429</b>	1.42
	H@10	0.0605	0.0610	0.0787	0.0805	<u>0.0822</u>	<b>0.0840</b>	2.19
	N@5	0.0281	0.0283	0.0311	0.0342	<u>0.0353</u>	<b>0.0358</b>	1.42
	N@10	0.0351	0.0355	0.0391	0.0413	<u>0.0428</u>	<b>0.0434</b>	1.40



**Fig. 5** Trade-off between prediction generation time and recommendation quality across multiple datasets. X-axis shows the time overhead relative to the Zero method. Y-axis presents the HR@10 metric relative to the Zero method



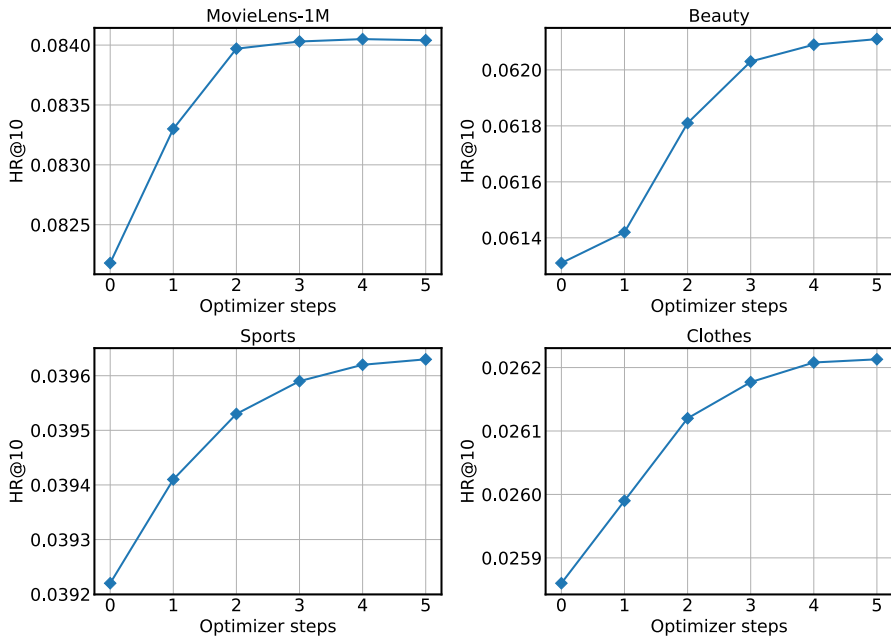
**Fig. 6** Comparison of model performance on the MovieLens-1M (left) and Amazon Beauty (right) datasets across varying embedding dimensions. Performance change is shown relative to MRGSRec+

while updating recommendations orders of magnitude faster than full model retraining (Fig. 5).

### 6.2 Effects of embedding size

To better understand the behavior of our method, we compare the performance of several models across a range of user and item embedding dimensions. For each dataset and model, we fix the embedding size and perform a grid search to tune the hyperparameters. The results are shown in Fig. 6, where we plot the performance change (in percentages) of each method relative to MRGSRec+.

As illustrated, *MRGSRec+* consistently outperforms both *MRGSRec* (Baikalov et al. 2024) and *SASRec+* (Mezentsev et al. 2024) across all embedding sizes and datasets. This confirms the robustness of our *Folding-In* procedure, which provides consistent performance gains in different configurations. Notably, *SASRec+* exhibits a



**Fig. 7** Comparison of models performance for different number of Folding-In optimizer steps

significant drop in performance at lower embedding dimensions, in contrast to the more stable behavior of the multi-representational models. This highlights the advantage achieved by the graph-based component in enhancing performance, especially in low-capacity regimes.

### 6.3 Computational complexity

Additionally, we analyze how the performance of MRGSRec+ changes with the number of optimization steps in the *Folding-In* procedure (see Fig. 7). In this experiment, we freeze the weights of the MRGSRec model and apply *Folding-In*, tuning the hyperparameter  $\lambda$  via a grid search. Across all datasets, no more than three optimization steps are needed to achieve optimal performance. As a result, applying the *Folding-In* procedure for all users adds less than 2.5% overhead to the original MRGSRec (Baikalov et al. 2024) training time. Compared to full model retraining, *Folding-In* is over 40 $\times$  faster. This makes our MRGSRec+ model significantly more efficient, saving both time and energy and making it more practical, cost-effective, and environmentally friendly.

To further demonstrate the efficiency of our method, we report both the total training time of the full model and the time required to apply the *Folding-In* procedure for a single user. The efficiency gain arises from avoiding full retraining. Incorporating a new user via retraining requires re-optimizing the entire model, whereas our *Folding-In* approach directly computes the new user embedding while keeping other parameters

**Table 3** Time durations (in seconds) for training and Folding-In procedures of the MRGSRec model across different datasets

Dataset	Training, s	Folding-In, s	Speedup, %
ML-1 M	4783	6.8	703
Beauty	5390	7.9	682
Clothing	7439	17.1	435
Sports	8067	17.8	453

fixed. As shown in Table 3, the Folding-In procedure for a single user achieves up to a  $700\times$  speed-up compared to full model retraining. This highlights the practicality of our approach to updating personal recommendations, offering a highly efficient solution that significantly reduces computational and energy costs.

## 7 Conclusion

In this work, we introduced the *Folding-In* procedure for updating the user embeddings in graph-based recommender models. This procedure enables the model to seamlessly incorporate new users into the system or update the representation of existing users who interacted with new items after the initial training—all in an independent manner. By incorporating Folding-In into the graph component of the MRGSRec model we evaluate its performance under robust data splitting strategy that prevents data leakage. The resulting model, **MRGSRec+** achieves up to **2% (NDCG@5 metric) recommendation quality improvement** compared to other graph- and sequential-based baselines. Furthermore, our experiments demonstrate that Folding-In is superior to full retraining, providing up to a  **$700\times$  speedup** over full model retraining with minimal computational overhead. These results highlight the proposed *Folding-In* as a promising strategy for enhancing the adaptability, scalability, and practical deployment of graph-based recommender systems in dynamic real-world environments.

**Author Contributions** Viacheslav Yusupov: Conceptualization, Implementation, Investigation, Writing  
Nikita Sukhorukov: Conceptualization, Investigation, Writing Evgeny Frolov: Conceptualization, Supervision

**Funding** The work was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the Russian Federation in accordance with the agreement 000000C313925P4E0002 and the agreement with HSE University № 139-15-2025-009.

**Data Availability** <https://anonymous.4open.science/t/Fold-in-MRGSRec-CED6/README.md>

## Declarations

**Conflict of interest** The authors declare no Conflict of interest.

## References

- Amari, S.-I.: Backpropagation and stochastic gradient descent method. *Neurocomputing* **5**(4–5), 185–196 (1993)
- Ba, J.L.: Layer normalization. arXiv preprint [arXiv:1607.06450](https://arxiv.org/abs/1607.06450) (2016)
- Baikalov, V., Frolov, E.: End-to-end graph-sequential representation learning for accurate recommendations. In: Companion proceedings of the ACM on web conference 2024, pp. 501–504 (2024)
- Bellogin, A., Castells, P., Said, A., Tikk, D.: Report on the workshop on reproducibility and replication in recommender systems evaluation (repsys). In: ACM SIGIR Forum, vol. 48, pp. 29–35 (2014). ACM New York, NY, USA
- Chen, J., Wang, C., Zhou, S., Shi, Q., Chen, J., Feng, Y., Chen, C.: Fast adaptively weighted matrix factorization for recommendation with implicit feedback. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 3470–3477 (2020)
- Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and Short Papers), pp. 4171–4186 (2019)
- Gogna, A., Majumdar, A.: A comprehensive recommender system model: improving accuracy for both warm and cold start users. *IEEE Access* **3**, 2803–2813 (2015)
- Gusak, D., Mezentsev, G., Oseledets, I., Frolov, E.: Rece: Reduced cross-entropy loss for large-catalogue sequential recommenders. In: Proceedings of the 33rd ACM international conference on information and knowledge management, pp. 3772–3776 (2024)
- Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. *ACM Trans Interactive Intell Systems (TIIS)* **5**(4), 1–19 (2015)
- He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in information retrieval, pp. 639–648 (2020)
- He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th international conference on world wide web, pp. 507–517 (2016)
- He, X., Zhang, H., Kan, M.-Y., Chua, T.-S.: Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval, pp. 549–558 (2016)
- Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: Proceedings of the 27th ACM international conference on information and knowledge management, pp. 843–852 (2018)
- Hidasi, B.: Session-based recommendations with recurrent neural networks. arXiv preprint [arXiv:1511.06939](https://arxiv.org/abs/1511.06939) (2015)
- Hoecker, A., Kartvelishvili, V.: Svd approach to data unfolding. *Nucl. Instrum. Methods Phys. Res., Sect. A* **372**(3), 469–481 (1996)
- Jing, M., Zhu, Y., Zang, T., Wang, K.: Contrastive self-supervised learning in recommender systems: a survey. *ACM Trans Inf Syst* **42**(2), 1–39 (2023)
- Kang, W.-C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International conference on data mining (ICDM), pp. 197–206 (2018). IEEE
- Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *CoRR* **abs/1412.6980** (2014)
- Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907) (2016)
- Klenitskiy, A., Vasilev, A.: Turning dross into gold loss: is bert4rec really better than sasrec? In: Proceedings of the 17th ACM conference on recommender systems, pp. 1120–1125 (2023)
- Klenitskiy, A., Volodkevich, A., Pembek, A., Vasilev, A.: Does it look sequential? an analysis of datasets for evaluation of sequential recommendations. In: Proceedings of the 18th ACM conference on recommender systems, pp. 1067–1072 (2024)
- Li, A., Yang, B., Huo, H., Hussain, F.K., Xu, G.: Self-supervised dual graph learning for recommendation. *Know.-Based Syst.* (2025). <https://doi.org/10.1016/j.knsys.2025.112967>
- Liang, S., Zhang, X., Ren, Z., Kanoulas, E.: Dynamic embeddings for user profiling in twitter. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 1764–1773 (2018)

- Mao, K., Zhu, J., Xiao, X., Lu, B., Wang, Z., He, X.: Ultracgn: ultra simplification of graph convolutional networks for recommendation. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp. 1253–1262 (2021)
- Mezentsev, G., Gusak, D., Oseledets, I., Frolov, E.: Scalable cross-entropy loss for sequential recommendations with large item catalogs. In: Proceedings of the 18th ACM conference on recommender systems, pp. 475–485 (2024)
- Peng, C., Xia, F., Naseriparsa, M., Osborne, F.: Knowledge graphs: opportunities and challenges. *Artif. Intell. Rev.* **56**(11), 13071–13102 (2023)
- Qiu, R., Huang, Z., Yin, H., Wang, Z.: Contrastive learning for representation degeneration problem in sequential recommendation. In: Proceedings of the fifteenth ACM international conference on web search and data mining, pp. 813–823 (2022)
- Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, pp. 452–461 (2009)
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
- Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp. 1441–1450 (2019)
- Sun, A.: Take a fresh look at recommender systems from an evaluation standpoint. In: Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval. SIGIR '23, pp. 2629–2638. Association for computing machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3539618.3591931>
- Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp. 1017–1024 (2011)
- Vaswani, A.: Attention is all you need. *Adv. Neural. Inf. Process. Syst.* **30**, 1 (2017)
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International conference on learning representations (2018)
- Wang, X., He, X., Wang, M., Feng, F., Chua, T.-S.: Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp. 165–174 (2019)
- Wang, Z., Liu, H., Wei, W., Hu, Y., Mao, X.-L., He, S., Fang, R., Chen, D.: Multi-level contrastive learning framework for sequential recommendation. In: Proceedings of the 31st ACM international conference on information & knowledge management, pp. 2098–2107 (2022)
- Wang, Y., Wang, L., Li, Y., He, D., Liu, T.-Y.: A theoretical analysis of ndcg type ranking measures. In: Conference on learning theory, pp. 25–54 (2013). PMLR
- Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., Xie, X.: Self-supervised graph learning for recommendation. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. SIGIR '21, pp. 726–735. Association for computing machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3404835.3462862>
- Wu, Q., Zhang, H., Gao, X., He, P., Weng, P., Gao, H., Chen, G.: Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In: The world wide web conference, pp. 2091–2102 (2019)
- Xin, D., Mayoraz, N., Pham, H., Lakshmanan, K., Anderson, J.R.: Folding: Why good models sometimes make spurious recommendations. In: Proceedings of the eleventh ACM conference on recommender systems, pp. 201–209 (2017)
- Yang, Z., Ding, M., Xu, B., Yang, H., Tang, J.: Stam: A spatiotemporal aggregation method for graph neural network-based recommendation. In: Proceedings of the ACM web conference 2022, pp. 3217–3228 (2022)
- Yang, L., Liu, B., Lin, L., Xia, F., Chen, K., Yang, Q.: Exploring clustering of bandits for online recommendation system. In: Proceedings of the 14th ACM conference on recommender systems, pp. 120–129 (2020)
- Zhang, W., Yang, L., Song, Z., Zou, H.P., Xu, K., Zhu, Y., Yu, P.S.: Sgcl: Unifying self-supervised and supervised learning for graph recommendation. In: Proceedings of the nineteenth ACM conference on recommender systems. RecSys '25, pp. 671–676. Association for computing machinery, New York, NY, USA (2025). <https://doi.org/10.1145/3705328.3748036>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

**Viacheslav Yusupov** is a researcher at HSE University and a M.Sc. student at HSE University. His work focuses on scalable methods and graph architectures in recommender systems. He holds a B.Sc. in Computer Science from HSE University.

**Nikita Sukhorukov** is a researcher at AIRI and a PhD student at Skoltech. His work focuses on methods for feature selection in recommender systems. He holds a M.Sc. in Computer Science from Skoltech and a B.Sc. in Physics from Moscow Institute of Physics and Technology.

**Evgeny Frolov** received the Graduate degree from Lomonosov Moscow State University, in 2009, and the Ph.D. degree from the Skolkovo Institute of Science and Technologies (Skoltech), in 2018, under the supervision of Ivan Oseledets. After graduation, he started developing career as an ICT Industry Professional, but returned to academia several years later. He leads the Technologies of Personalization group at AIRI, where he manages both academic and industrial research projects. His research is concentrated around building better bridges between practical challenges arising in the field of recommender systems and theoretical advances in relevant mathematical disciplines. He is especially attracted by algebraic and geometric methods, but also has broader interests. Some of his work is published at the leading ACM Recommender Systems Conference (RecSys). He is also the coauthor of a comprehensive survey on tensor methods in recommender systems.