♥On the Effectiveness of Unlearning in Session-Based Recommendation

Xin Xin*
Shandong University
Qingdao, China
xinxin@sdu.edu.cn

Liu Yang*
Shandong University
Qingdao, China
yangliushirry@gmail.com

Ziqi Zhao
Shandong University
Qingdao, China
ziqizhao.work@gmail.com

Pengjie Ren
Shandong University
Qingdao, China
jay.ren@outlook.com

Zhumin Chen
Shandong University
Qingdao, China
chenzhumin@sdu.edu.cn

Jun Ma
Shandong University
Qingdao, China
majun@sdu.edu.cn

Zhaochun Ren†
z.ren@liacs.leidenuniv.nl
Leiden University
Leiden, Netherlands

ABSTRACT

会话型推荐通过分析用户在会话中的先前交互来预测其未来的兴趣。尽管通过记忆历史样本可以实现这一目标，但有时也需要进行反学习，即移除特定训练样本的影响，原因可能是用户隐私或模型忠实度。然而，现有关于反学习的研究并未专门针对会话型推荐进行定制。一方面，由于反学习项与会话中其余项之间存在协同相关性和顺序连接，这些方法无法实现令人满意的反学习效果。另一方面，在会话型推荐场景中，很少有研究验证反学习的有效性。

在这篇论文中，我们提出了 SRU，一个面向会话型推荐的反学习框架，该框架实现了高效的反学习、准确的推荐性能以及改进的反学习效果。具体而言，我们首先根据会话之间的相似性将训练会话划分为不同的子模型，然后利用基于注意力的聚合层根据会话与子模型中数据质心之间的相关性来融合隐藏状态。为了提高反学习效果，我们进一步提出了三种额外数据删除策略，包括协同额外删除（CED）、邻近额外删除（NED）和随机额外删除（RED）。此外，我们提出了一个评估指标，用于衡量在数据删除后是否能够推断出反学习样本，以验证反学习的有效性。我们使用三个代表性的会话型推荐模型实现了 SRU，并在三个基准数据集上进行了实验。实验结果表明我们方法的有效性。代码和数据可在 https://github.com/shirryliu/SRU-code 获取。

1 INTRODUCTION

会话型推荐模型已经展现出在预测用户未来兴趣方面的有效性，通过记忆的顺序交互[16, 41]。然而，消除特定训练样本的影响，即反学习，也具有重要意义。从合法性的角度来看，已经提出了一些数据保护法规，例如《通用数据保护条例》（GDPR）[26]和《加利福尼亚消费者隐私法案》（CCPA）[17]。这些法规强调个人有权要求将其私人信息从训练好的机器学习模型中删除。从用户的角度来看，已经涌现出一系列研究证明，从与推荐系统的历史交互中可以推断出各种用户隐私信息，如性别、年龄，甚至政治取向[4, 6, 43]。针对隐私问题，用户可能会认为请求删除特定的历史交互是至关重要的。此外，一种高效的推荐模型具备消除嘈杂训练交互影响的能力，以获得更好的性能。

机器反学习。机器反学习使模型能够忘记先前学到的某些数据或模式。精确的反学习旨在完全消除要遗忘的数据的影响，就像它们在训练过程中从未发生过一样。一种直接的精确反学习方法是从训练数据集中移除目标样本，然后从头开始重新训练整个模型。不幸的是，这种方法因为耗时和资源密集而受到阻碍。为了解决这个问题，现有的方法[1, 2]着重于提高反学习的效率。其中最具代表性的反学习方法之一是SISA[1]。SISA首先将训练数据集分成大小相等的不相交分片。随后，在每个分片上独立训练子模型。为了制定给定数据点的最终模型预测，通过多数投票或平均值将每个子模型的预测聚合起来。在发生反学习请求时，只需重新训练在包含反学习数据点的分片上训练的子模型，而不是整个模型。与整体重新训练相比，SISA在反学习效率方面取得了显著的改进。

在会话型推荐中反学习面临的挑战：

i)．难以实现精确的反学习。现有的精确反学习方法基于这样一个假设：如果在重新训练的模型中不存在反学习样本，那么反学习样本的影响将完全被消除。然而，这个假设在会话型推荐中并不成立。与图像分类等其他领域不同，在会话型推荐中，交互项之间存在大量的协同相关性和顺序连接，而不是稀疏的。因此，简单地移除反学习样本无法实现精确的反学习效果，即从会话中剩余的项中仍然可以推断出已反学习的项，如图1所示。

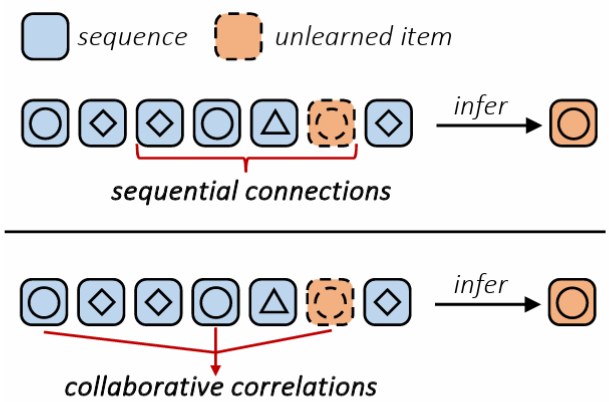

Figure 1: Exact unlearning is hard to achieve. The unlearned item could still be inferred due to collaborative correlations and sequential connections across items in the session.

ii). 现有的推荐反学习方法没有评估反学习效果。现有方法[5, 23]主要关注推荐性能与反学习效率之间的权衡。然而，很少有研究对反学习效果进行评估，即在多大程度上消除了反学习样本的影响。对于会话型推荐，评估尤其重要，考虑到精确反学习无法简单实现的情况。

为了解决第一个挑战，我们提出了三种额外数据删除策略，包括协同额外删除（CED）、邻近额外删除（NED）和随机额外删除（RED），以进一步提高反学习的效果。对于第二个挑战，我们提出了一个评估指标，用于衡量在数据删除后是否能够推断出反学习样本。直觉是，如果反学习非常有效，那么基于剩余数据推断反学习样本的概率应该很低。当发生反学习请求时，例如用户可能希望隐藏对敏感项的点击，只需要重新训练相应的子模型和聚合层，基于删除的数据实现高效的反学习。

为了验证所提出方法的有效性，我们在三个代表性的会话型推荐模型上实施了SRU，包括GRU4Rec [15]、SASRec [20]和BERT4Rec [31]。我们在三个基准数据集上进行了一系列实验，结果显示了所提方法的有效性。

贡献总结如下：
• 据我们所知，SRU是首次尝试解决会话型推荐中的机器反学习问题。我们提出了三种额外数据删除策略以提高反学习效果，同时利用基于相似性的聚类和基于注意力的聚合来保持高推荐性能。
• 我们提出了一个评估指标，用于验证会话型推荐的反学习效果。其核心思想是，如果反学习有效，那么在数据删除后，反学习样本被推断出的概率应该很低。
• 我们在三个最先进的会话型推荐模型和三个基准数据集上进行了大量实验证明，SRU能够在保持高推荐性能的同时实现高效且有效的反学习。

2 RELATEDWORK

2.1 会话型推荐

会话型推荐旨在从用户在会话中的过去交互中捕获其动态兴趣。早期基于马尔可夫链的模型 [12, 13, 27, 29] 根据给定会话中的最后一次交互预测用户即将出现的兴趣。近年来，利用深度神经网络模型来捕捉复杂的序列信号以改进会话型推荐的效果。代表性的会话型推荐模型可以分为基于循环神经网络（RNN）的模型 [9, 14]、基于卷积神经网络（CNN）的模型 [32]、基于注意力的模型 [20, 31] 和基于图的模型 [36]。此外，自监督学习 [42] 和对比学习 [24, 37] 也已应用于改进会话型推荐，涌现出了许多模型。

在本文中，我们提出了一个框架，使各种会话型推荐模型能够进行有效且高效的反学习，而不是开发一个新的具体模型。我们选择了三个代表性的模型，GRU4Rec、SASRec 和 BERT4Rec，作为实验的基础模型。

2.2 机器反学习

机器反学习的概念最早由[3]提出，以响应“被遗忘的权利”的要求。反学习方法大致可分为近似反学习方法和精确反学习方法。

近似反学习确保了反学习后的模型性能与重新训练模型的性能密切一致。这降低了反学习的时间和计算成本，但可能会牺牲一定程度的隐私保证。这种近似可以通过差分隐私技术来实现，例如认证反学习[39]。例如，[33]提出了一种基于噪声随机梯度下降的反学习方法，而[10]则基于牛顿更新实现了认证反学习。[3]提出使用梯度手术，使用反学习样本的负梯度更新模型参数。[18]利用概率模型来近似反学习过程。[7, 25]提出通过逆Hessian矩阵扰动梯度或模型权重，这可能会带来额外的计算开销。

精确反学习旨在完全消除反学习样本的影响，就像它们在训练过程中从未发生过一样，提供更强的隐私保证。然而，这种方法可能要求从头开始重新训练模型，这在计算上是昂贵且耗时的。对于高效的精确反学习，最具代表性的方法是SISA [1]，因为只有在相应的数据分片上训练的子模型会在反学习请求时重新训练。[8]将SISA方法改编用于图神经网络的反学习。[11]修改了SISA算法，以适用于删除请求的序列。另一种精确反学习的方法涉及选择性影响估计器 [35]，它们计算反学习样本对模型参数的影响。尽管这些基于影响的方法在隐私保护方面有效，但高计算成本限制了它们在实际场景中的应用 [39]。

最近，在推荐场景中的反学习越来越受到研究关注。反学习不仅有助于保护用户隐私，还通过消除嘈杂数据和误导性信息的影响来改善推荐模型 [28]。[23]和[40]提出使用微调和最小二乘算法来加速反学习。[5]和[22]扩展了SISA算法的思想，用于协同过滤。然而，目前尚无现有方法专门为会话型推荐定制。此外，现有方法主要关注反学习的效率，而未能验证反学习的有效性，即在多大程度上消除了反学习样本的影响。

3 TASKFORMULATION

在本节中，我们首先对会话型推荐的任务进行阐述，随后定义了会话中项目级别反学习的任务。然后，我们识别了反学习面临的挑战。

3.1 符号和定义

会话型推荐的目标是根据用户在会话中之前与项目的交互来预测用户可能的下一步行为。我们将任务形式化如下：

Definition 3.1 (Session-based recommendation). Let $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ be the set of items, $\mathcal{D}$ denotes the training interaction sessions. $\mathcal{S}_i = [v_1^i, ..., v_t^i, ..., v_n^i] \in \mathcal{D}$ denotes the i-th specific interaction session in $\mathcal{D}$, where $v_t^i \in \mathcal{V}$ is the item interacted by the user at time step t, and n is the current length of the session. Given the historical sequence $\mathcal{S}_i$, the interaction probability over candidate item v at time step $n+1$ can be formalized as:

$$p_v^i = p(v_{n+1}^i = v|\mathcal{S}_i, \mathcal{D}) = \mathcal{M}(v|\mathcal{S}_i, \mathcal{D}), \tag{1}$$

where $\mathcal{M}$ denotes the involved recommendation model, e.g., GRU4Rec [14] and SASRec [20]. At the prediction stage, session-based recommenders select the items with the highest top-K probability p_v^i as the recommendation list for the user.

出于隐私考虑或推荐效用的原因，可能会发生反学习请求，以消除某些训练样本的影响。 举例来说，用户可能希望撤销在交互会话中的一些误点，因为误点可能降低推荐的质量，或者用户也可能出于隐私考虑要求隐藏某些敏感项目的点击。在本文中，我们关注会话型推荐中的项目级别反学习，其定义如下：

Definition 3.2 (Item-level unlearning). We denote $v_j^i \in \mathcal{S}_i$ to be the unlearning item that the user wants to revoke in the session $\mathcal{S}_i$. The goal of item-level unlearning is to obtain an unlearned model $\mathcal{M}_u$. Ideally, the unlearning sample v_j^i should have no effect on the unlearned model $\mathcal{M}_u$ as if v_j^i never occurred in the session.

请注意，除了项目级别的反学习外，还可能存在对会话级别反学习的请求，即消除整个交互会话的影响。在本文中，我们专注于项目级别的反学习，尽管所提出的框架也可以支持会话级别的反学习。我们将对会话级别反学习的进一步研究留作未来的方向之一。

3.2 Challenges

3.2.1 实现精确反学习非常困难。 现有的精确反学习方法主要应用于诸如图像分类 [1] 等领域，其中数据点相对独立。在这种情况下，现有的反学习方法认为，如果在重新训练的模型中不存在反学习样本，那么反学习样本的影响将被完全消除。然而，由于在基于会话的推荐中存在大量协同相关性和顺序连接的项目交互，简单地删除反学习样本无法达到期望的效果，即使在训练数据中不存在反学习样本。例如，用户可能希望在会话中隐藏对敏感项目的点击，而仅删除会话中的反学习项目无法达到满意的效果，因为由于协同相关性或顺序相关性的存在，仍然可以从删除的数据中推断出敏感项目。这一挑战可以形式化为

$$\text{exact unlearning} \neq \mathcal{M}_u(v|\mathcal{S}_i', \mathcal{D}\backslash\mathcal{S}_i \cup \mathcal{S}_i'), \text{where } \mathcal{S}_i' = \mathcal{S}_i \backslash v_j^i. \tag{2}$$

3.2.2 反学习效果缺乏明确定义。 现有的推荐反学习方法 [5, 23] 主要研究了推荐性能和反学习效率之间的权衡。正如在第3.2.1节中指出的，会话型推荐中很难实现精确反学习。在这种情况下，对反学习效果的评估，即消除反学习样本的影响程度，显得尤为重要。然而，在会话型推荐领域，很少有工作对反学习效果进行评估。

尽管在会话型推荐中存在上述两个具体的反学习挑战，但模型性能（即推荐准确性）和反学习效率也是需要优化的关键因素。

在这一部分，我们详细描述了提出的SRU框架。 如图2所示，SRU由会话划分、注意力汇聚和数据删除组成。会话划分模块旨在将训练会话划分为不相交的数据分片，然后在每个分片上训练子模型。基于来自不同子模型的隐藏状态，注意力汇聚模块融合这些隐藏状态以进行最终预测。数据删除模块旨在提高反学习的效果。当出现项目级别的反学习请求时，数据删除模块首先应用额外的数据删除策略到相应的会话中。然后仅重新训练子模型和聚合模块，实现高效的反学习。

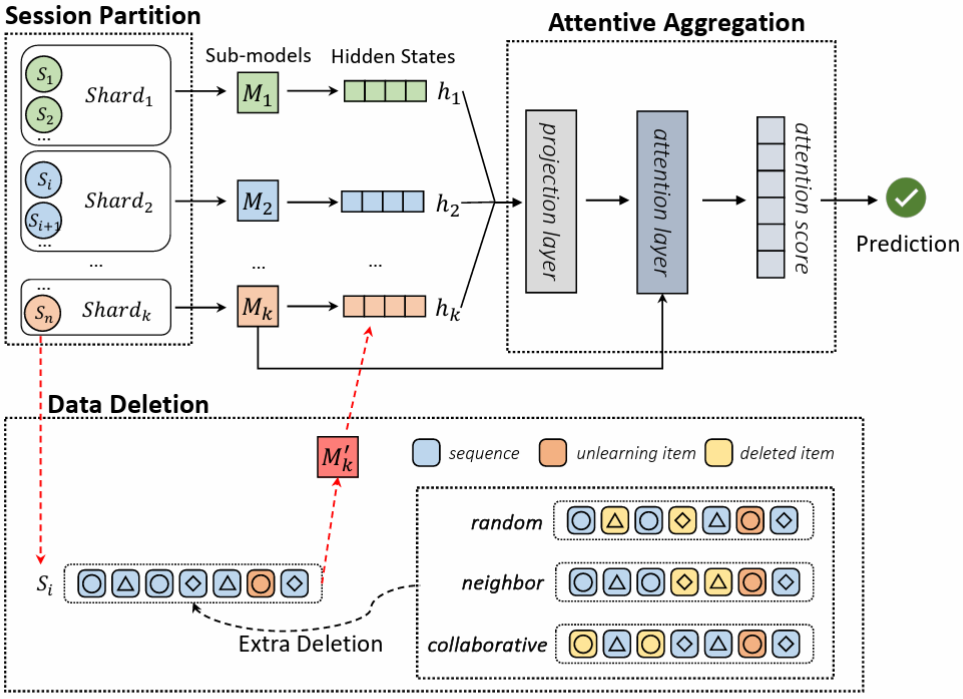

Figure 2: Overview of the proposed SRU framework. SRU is composed of session partition, attentive aggregation and data deletion modules.

4 METHODOLOGY

4.1 会话划分

生成下一个项目推荐的关键之一是从相似会话中学习信号。为此，会话的相似性对于推荐准确性至关重要。因此，在会话划分模块中，预期相似的会话将被划分到相同的数据分片中，因此可以在一个子模型中进行训练。这样的划分策略有助于提高推荐性能，因为它在每个分片内实现更多的知识传递。

为实现上述划分策略，首先在数据集D上预先训练一个附加的基于会话的推荐模型Mp（例如，GRU4Rec[15]），以获取所有训练会话的隐藏状态。然后，基于预先训练的隐藏状态使用k均值聚类方法来划分训练会话。具体来说，会话划分模块的输入包括预先训练的隐藏状态、划分分片数K和每个分片中最大的会话数δ。会话对的距离定义为它们隐藏状态的欧几里德距离。首先随机选择K个会话作为初始质心，然后计算会话和质心之间的距离。随后，根据距离的升序顺序，将会话顺序分配到分片中。如果一个分片不可用（即分片内的会话数大于δ），则将下一个会话分配到最近可用的分片。之后，计算每个对应分片中所有会话的隐藏状态的平均值作为新的质心。重复以上过程直到质心不再更新。然后，我们获得了平衡的划分会话，即 $\bigcap_{k\in[\mathcal{K}]}\mathcal{D}_k = \varnothing$ and $\bigcup_{k\in[\mathcal{K}]}\mathcal{D}_k = \mathcal{D}$。然后，分别在每个数据分片上训练子模型。

4.2 注意力聚合

基于会话划分，每个子模型倾向于学习相似的顺序模式的聚类。注意力聚合模块的目标是融合来自每个子模型的隐藏状态进行最终预测，该模块包括一个投影层、一个注意力层和一个输出层。

投影层：即全连接层

注意力层：即注意力层

输出层：也是全连接层

4.3 DataDeletion

数据删除模块旨在提高反学习的效果。 对于项目级别的反学习请求，传统的反学习方法只是删除反学习样本，但仍然可能通过会话中剩余的交互来推断已删除的样本，因为存在顺序连接和协同相关性。为解决这个问题，我们提出了三种策略，即协同额外删除（CED）、邻居额外删除（NED）和随机额外删除（RED）。

从协同相关性的角度来看，我们提出了CED，它根据反学习项目与会话中其他项目之间的相似性删除额外的项目。给定会话Si中的目标反学习项目vij，根据从预先训练的模型Mp获取的项目嵌入计算项目相似性的欧氏距离。之后，按距离的升序对会话中的项目进行排序，并从会话中删除最相似的N个项目。最后，重新训练相应的子模型和聚合模块。反学习模型可以形式化为：

$$\mathcal{M}_u(v|\mathcal{S}_i'', \mathcal{D}\backslash\mathcal{S}_i \cup \mathcal{S}_i''), \text{where } \mathcal{S}_i'' = \mathcal{S}_i\backslash\text{CED}(v_j^i). \qquad (7)$$

至于顺序连接，我们提出了NED，它用于按时间顺序删除在反学习项目前的N个最近项目。而在RED中，我们随机选择N个额外项目在会话中删除。

4.4 反学习效果评估

项目级别的反学习是一种常见请求，例如，用户可能希望在会话中隐藏对敏感项目的点击，或者可能不再喜欢某个项目。为此，如果反学习是有效的，那么反学习项目不应该从会话中剩余的项目中推断出，也不应该在近期再次向用户推荐。

为此，我们定义了一种反学习效果评估指标，即命中率（HIT@K），该指标衡量了在使用反学习模型M_u基于会话中的剩余交互生成的前K个推荐列表中，反学习项目是否出现。这样的评估指标也可以被视为成员推理攻击 [30] 的性能，该攻击试图从剩余数据中推断出反学习项目。如果HIT@K较高，则意味着反学习项目有很高的概率被重新推荐或再次推断。相反，较低的HIT@K意味着反学习模型M_u已经很好地反学习了项目，并且达到了更好的反学习效果。

5 EXPERIMENTS

在本节中，我们在三个基准数据集上进行实验证明SRU的有效性。我们的目标是回答以下研究问题：

- **RQ1：当使用不同的基于会话的推荐模型实例化SRU时，SRU的推荐性能如何?**
- **RQ2：SRU的反学习效果如何?**
- **RQ3：SRU的反学习效率如何?**

关于SRU设计如何影响性能的更多消融实验可以参见附录。

5.1 实验设置

5.1.1 数据集。 实验使用三个公开可访问的数据集进行，分别是Amazon Beauty、Games和Steam。这两个Amazon数据集是从Amazon.com抓取的一系列产品评论数据集。在这项工作中，我们考虑了两个商品类别，分别是“Beauty”和“Games”。Steam数据集是从一个大型在线视频游戏分发平台收集的。对于所有数据集，我们遵循与[20]相同的数据预处理流程。表1显示了数据集的统计信息。

Table 1: Statistics of the datasets (after preprocessing).

Dataset	#users	#items	#actions
Amazon Beauty	52,024	57,289	0.4M
Amazon Games	31,013	23,715	0.3M
Steam	334,730	13,047	3.7M

5.1.2 推荐性能评估。 我们采用交叉验证来评估提出方法的性能。训练、验证和测试集的比例为8:1:1。我们随机抽样80%的会话作为训练集。对于验证和测试集，通过逐个提供会话中的交互并检查下一个真实项目的排名来进行评估。排名是在整个项目集中进行的。

为了评估推荐性能，我们采用两个常见的Top-K指标：Recall@K 和 NDCG@K。Recall@K 衡量了真实项目是否在推荐列表的前K位置上[38]。NDCG@K 是一种加权指标，将更高的分数分配给排名靠前的位置[19]。我们使用在第4.4节中描述的HIT指标来评估反学习效果。

基于会话的推荐模型： GRU4Rec[15]、SASRec[20] 和 BERT4Rec [31]。

- **GRU4Rec[15]：** 该方法利用门控循环单元（GRU）来建模用户交互序列。
- **SASRec [20]：** 该模型是基于注意力机制的，使用Transformer [34]解码器进行基于会话的推荐。
- **BERT4Rec [31]：** 该模型使用深度双向自注意力来建模交互序列。

为了实现反学习，每个模型都经过以下训练：

- **Retrain：** 该方法从头开始对剩余数据集进行整个模型的重新训练。计算成本较高。
- **SISA：** 这是一种基本的精确反学习方法，随机分割数据并平均子模型的输出。
- **SRU-N：** 这是带有邻近额外删除（NED）的SRU。
- **SRU-R：** 这是带有随机额外删除（RED）的SRU。
- **SRU-C：** 这是带有协作额外删除（CED）的SRU。

请注意，我们不与RecEraser[5]进行比较，因为它是为非顺序协同过滤提出的，而他们的数据分割方法无法应用于基于会话的推荐，因为基于会话的推荐器不明确地建模用户标识符。

5.1.4 超参数设置。 模型输入是Beauty的最后10个互动项目，Games和Steam的最后20个互动项目。我们对较短的会话使用填充标记进行填充。使用Adam优化器[21] 对所有模型进行训练，批次大小为256。聚合层的学习率在 [1e-3, 1e-2] 范围内进行调整。默认的数据分片数目设置为K = 8。用于反学习的额外数据删除数量范围从1到5。其他超参数设置为它们原始论文中的推荐设置。

5.2 推荐性能 (RQ1)

表2显示了不同反学习方法在每个分片中需要取消学习10%的随机会话中的Top-K推荐性能。我们可以看到，即使SRU删除了更多的训练数据，但它总是比SISA表现更好。这是因为当训练数据被取消学习时，所有子模型的性能都会下降，因为训练数据更少，而SRU将相似的会话分组在一个分片中，使模型共享更多的协作信息以获得更好的推荐性能。并且Retrain总是获得最高的分数，因为它可以在所有剩余数据上重新训练整个模型，但牺牲了效率。

Table 3: Recommendation performance comparison without unlearning request. SRU denotes the proposed SRU framework without extra data deletion. Best results other than Retrain are highlighted in bold. "N" is short for NDCG and "R" is short for Recall.

Beauty	GRU4Rec				SASRec				BERT4Rec			
	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20
Retrain	0.0340	0.0396	0.0580	0.0801	0.0419	0.0471	0.0668	0.0871	0.0366	0.0430	0.0634	0.0889
SISA	0.0309	0.0347	0.0490	0.0642	0.0283	0.0323	0.0453	0.0614	0.0271	0.0323	0.0491	0.0694
SRU	**0.0313**	**0.0360**	**0.0507**	**0.0691**	**0.0296**	**0.0341**	**0.0477**	**0.0655**	**0.0316**	**0.0371**	**0.0558**	**0.0777**
Steam	**GRU4Rec**				**SASRec**				**BERT4Rec**			
	N@10	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**
Retrain	0.0490	0.0623	0.0943	0.1475	0.0541	0.0682	0.1021	0.1584	0.0613	0.0764	0.1142	0.1742
SISA	0.0473	0.0600	0.0903	0.1412	0.0463	0.0586	0.0874	0.1366	0.0492	0.0628	0.0947	0.1489
SRU	**0.0474**	**0.0603**	**0.0903**	**0.1415**	**0.0483**	**0.0615**	**0.0915**	**0.1437**	**0.0575**	**0.0718**	**0.1072**	**0.1643**
Games	**GRU4Rec**				**SASRec**				**BERT4Rec**			
	N@10	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**
Retrain	0.0416	0.0516	0.0778	0.1174	0.0493	0.0604	0.0877	0.1319	0.0471	0.0595	0.0903	0.1397
SISA	0.0330	0.0389	0.0561	0.0795	0.0291	0.0346	0.0503	0.0723	0.0329	0.0414	0.0639	0.0974
SRU	**0.0370**	**0.0443**	**0.0642**	**0.0930**	**0.0348**	**0.0425**	**0.0632**	**0.0943**	**0.0419**	**0.0524**	**0.0802**	**0.1219**

此外，我们还进行了实验，观察当没有学习请求时的推荐性能。表3显示了在完整训练数据的情况下的推荐性能。Retrain模型获得了最佳分数。与SISA相比，我们可以看到提出的SRU总是获得更好的推荐性能。例如，在Steam数据集上训练的BERT4Rec模型中，SRU的NDCG@10为0.0575，而SISA的对应结果为0.0492，实现了16.9%的改进。这一观察证明了SRU的分区会话和关注聚合的有效性。

Table 3: Recommendation performance comparison without unlearning request. SRU denotes the proposed SRU framework without extra data deletion. Best results other than Retrain are highlighted in bold. "N" is short for NDCG and "R" is short for Recall.

Beauty	GRU4Rec				SASRec				BERT4Rec			
	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20
Retrain	0.0340	0.0396	0.0580	0.0801	0.0419	0.0471	0.0668	0.0871	0.0366	0.0430	0.0634	0.0889
SISA	0.0309	0.0347	0.0490	0.0642	0.0283	0.0323	0.0453	0.0614	0.0271	0.0323	0.0491	0.0694
SRU	**0.0313**	**0.0360**	**0.0507**	**0.0691**	**0.0296**	**0.0341**	**0.0477**	**0.0655**	**0.0316**	**0.0371**	**0.0558**	**0.0777**
Steam	**GRU4Rec**				**SASRec**				**BERT4Rec**			
	N@10	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**
Retrain	0.0490	0.0623	0.0943	0.1475	0.0541	0.0682	0.1021	0.1584	0.0613	0.0764	0.1142	0.1742
SISA	0.0473	0.0600	0.0903	0.1412	0.0463	0.0586	0.0874	0.1366	0.0492	0.0628	0.0947	0.1489
SRU	**0.0474**	**0.0603**	**0.0903**	**0.1415**	**0.0483**	**0.0615**	**0.0915**	**0.1437**	**0.0575**	**0.0718**	**0.1072**	**0.1643**
Games	**GRU4Rec**				**SASRec**				**BERT4Rec**			
	N@10	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**
Retrain	0.0416	0.0516	0.0778	0.1174	0.0493	0.0604	0.0877	0.1319	0.0471	0.0595	0.0903	0.1397
SISA	0.0330	0.0389	0.0561	0.0795	0.0291	0.0346	0.0503	0.0723	0.0329	0.0414	0.0639	0.0974
SRU	**0.0370**	**0.0443**	**0.0642**	**0.0930**	**0.0348**	**0.0425**	**0.0632**	**0.0943**	**0.0419**	**0.0524**	**0.0802**	**0.1219**

总的来说，提出的SRU在取消学习请求场景和完整数据场景下均实现了比基线方法更好的推荐性能。

5.3 取消学习效果（RQ2）

在这一部分，我们进行实验来评估不同方法的取消学习效果。我们随机取消10%的数据，并将额外删除数量N设置为1到5。表4显示了Beauty和Steam数据集上的取消学习效果比较。Games数据集的结果也显示了类似的结论。

首先，我们可以看到即使取消学习项目被移除，仍然存在一定的概率（例如，在Steam数据集上超过10%）可以从会话中的剩余交互中再次推断该项目。这一观察验证了传统的精确取消学习方法在基于会话的推荐场景中无法实现精确取消学习效果。

此外，我们可以看到提出的SRU-R、SRU-C和SRU-N相较于Retrain和SISA实现了更好的取消学习效果。例如，在GRU4Rec模型上，对于在Beauty数据集上进行训练，SRU的HIT@1为0.0577，而Retrain为0.0764。这一观察表明了提出的数据删除模块对取消学习效果至关重要。

更重要的是，SRU-C和SRU-N在取消学习效果方面取得了稳定的改进，因为它们可以帮助消除协作关系和顺序连接的影响，而SRU-R则随机删除额外数据，具有更加多样化的性能。

总的来说，提出的SRU实现了最高的取消学习效果，甚至优于Retrain。

Table 4: Unlearning effectiveness comparison. Lower scores denote better results. The best results are highlighted in bold.

Beauty	GRU4Rec				SASRec				BERT4Rec			
	HIT@1	HIT@5	HIT@10	HIT@20	HIT@1	HIT@5	HIT@10	HIT@20	HIT@1	HIT@5	HIT@10	HIT@20
Retrain	0.0764	0.1715	0.2294	0.3052	0.0619	0.1566	0.2123	0.2807	0.0700	0.1588	0.2080	0.2739
SISA	0.0685	0.1654	0.2244	0.3074	0.0681	0.1605	0.2222	0.3091	0.0763	0.1730	0.2321	0.3119
SRU-R	0.0675	0.1561	0.2122	0.2809	0.0625	0.1468	0.2042	0.2697	0.0720	0.1573	0.2131	0.2798
SRU-C	**0.0577**	**0.1335**	**0.1824**	**0.2510**	**0.0593**	**0.1429**	**0.1970**	**0.2666**	0.0661	**0.1516**	0.2058	**0.2689**
SRU-N	0.0643	0.1533	0.2028	0.2731	0.0605	0.1482	0.2039	0.2736	**0.0638**	0.1527	**0.2054**	0.2759
Steam	**GRU4Rec**				**SASRec**				**BERT4Rec**			
	HIT@1	HIT@5	HIT@10	HIT@20	HIT@1	HIT@5	HIT@10	HIT@20	HIT@1	HIT@5	HIT@10	HIT@20
Retrain	0.1581	0.3992	0.5372	0.6805	0.1411	0.3636	0.4975	0.6483	0.1159	0.3292	0.4701	0.6309
SISA	0.1582	0.3979	0.5349	0.6775	0.1410	0.3646	0.4959	0.6365	0.1166	0.3282	0.4668	0.6184
SRU-R	0.1545	0.3954	0.5319	0.6739	0.1412	0.3687	0.5020	0.6417	0.0992	0.2979	0.4282	0.5749
SRU-C	0.1499	0.3882	0.5241	0.6702	0.1389	0.3686	0.5041	0.6475	0.1036	0.3088	0.4407	0.5901
SRU-N	**0.1461**	**0.3799**	**0.5136**	**0.6568**	**0.1138**	**0.3186**	**0.4422**	**0.5812**	**0.0957**	**0.2897**	**0.4205**	**0.5713**

5.4 取消学习效率 (RQ3)

表5显示了Retrain和SRU之间的训练时间比较。我们在NVIDIA GeForce RTX 2080 Ti上评估它们，并将shard数量设置为8。特别是，SRU的重新训练时间包括子模型训练和聚合模块训练。

从表5中，我们可以看到SRU的效率远高于Retrain。在大多数情况下，SRU的速度比Retrain快三倍以上。例如，在Beauty和BERT4Rec上，Retrain需要55.76分钟，而我们的SRU只需12.97分钟。在较大的Steam数据集上，效率提升更为显著。例如，在Steam和SASRec上，Retrain需要368.99分钟，而我们的SRU只需99.31分钟，改进达到了3.71倍的优化。子模型的训练可以并行进行，因为它们不共享参数，这可以进一步加速训练过程。

Table 5: Comparison of unlearning efficiency (minute [m]). The best results are highlighted in bold.

Dataset		Beauty			Games			Steam		
Method		GRU4Rec	SASRec	BERT4Rec	GRU4Rec	SASRec	BERT4Rec	GRU4Rec	SASRec	BERT4Rec
Retrain		46.80	55.6	55.76	31.22	29.91	31.14	274.67	368.99	296.89
SRU	**Sub-model**	5.80	5.07	7.44	3.76	4.75	4.80	33.67	36.78	34.07
	Aggregation	0.72	6.05	5.53	1.78	4.40	3.87	25.30	62.53	64.30
	Total	**6.52**	**11.12**	**12.97**	**5.54**	**9.15**	**8.67**	**58.97**	**99.31**	**98.37**

6 CONCLUSION

在这篇论文中，我们提出了一个面向模型的遗忘框架SRU，用于基于会话的推荐。对于项目级别的遗忘请求，SRU利用三种数据删除策略，包括协同额外删除（CED），邻居额外删除（NED）和随机额外删除（RED），以确保无法再从会话中的剩余项目中推断出已遗忘的项目。然后，我们重新训练相应的子模型和聚合模块以实现高效的遗忘。我们利用基于相似性的会话分区模块和关注聚合模块来提高SRU的推荐性能。此外，我们进一步定义了一个评估指标来验证基于会话的推荐的遗忘效果。我们使用三个代表性的基于会话的推荐模型实现了SRU，并从遗忘性能、效率和效果的角度在三个基准数据集上进行了实验。实验结果显示了我们提出的方法的优越性。

对于未来的工作，我们计划研究基于会话的遗忘。在现实世界的推荐场景中，需要考虑会话级和项目级的遗忘，并且它们可能面临不同的挑战。此外，我们希望扩展确切的删除策略，以实现更精确的性能和完全的遗忘。我们还计划将该方法适应到其他推荐领域。此外，遗忘效果、推荐性能和遗忘效率之间的权衡也是一个有趣的未来研究课题。

A APPENDIX

A.1 消融研究

A.1.1 分片数量

在这一部分，我们进行实验来调查分片数量的影响。图3显示了不同分片数量下的NDCG@20和遗忘时间成本的结果。我们可以看到，较大的分片数量导致了推荐性能的降低和较低的遗忘时间成本。由于子模型是分别训练的，较大的分片数量表示跨会话的相关性较少，导致了较低的推荐性能。因此，需要一个良好的聚合层来整合子模型的信息。在推荐性能和遗忘成本之间的权衡是一个有趣的研究方向。

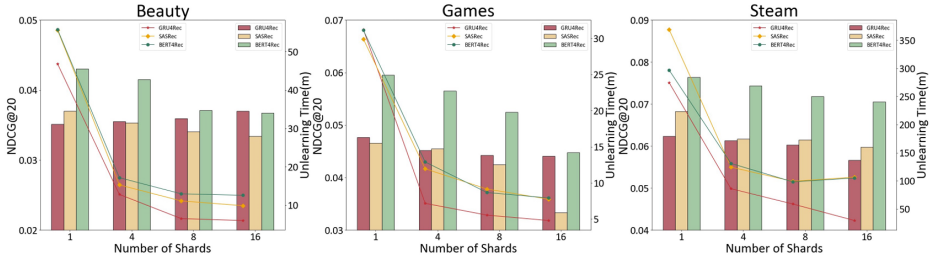

Figure 3: Impact of the shard number on recommendation performance and unlearning efficiency. The bar shows recommendation performance and the line shows unlearning time cost.

A.1.2 会话分割

在这一部分，我们进行实验来验证会话分割模块的效果。图4展示了在三个数据集上，使用提出的分割方法和随机分割方法的情况下，推荐性能（Recall@20）的结果。我们可以观察到，与随机分割方法相比，提出的方法实现了更好的推荐性能。例如，在Beauty数据集上，使用GRU4Rec模型的SRU的Recall@20为0.069，而相应的SISA的Recall@20为0.061。这一观察结果表明，提出的会话分割模块有助于提高推荐性能。

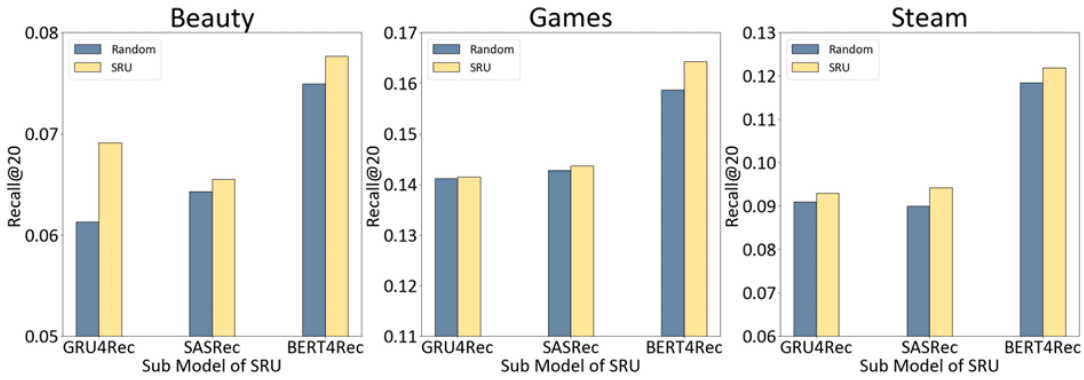

Figure 4: Effect of data partition.

A.1.3 数据删除

在这一部分，我们进行实验来研究额外删除的样本数量如何影响未学习的效果。我们将额外删除数量N从0变化到5。图5展示了在Games数据集上的结果。其他数据集的结果呈现类似的趋势。我们可以看到，随着额外删除数量的增加，从剩余数据中推断未学习项的概率减小。直观地说，增加的额外删除数量也可能降低推荐性能。在实际应用中，需要更深入地研究未学习效果和推荐性能之间的平衡。

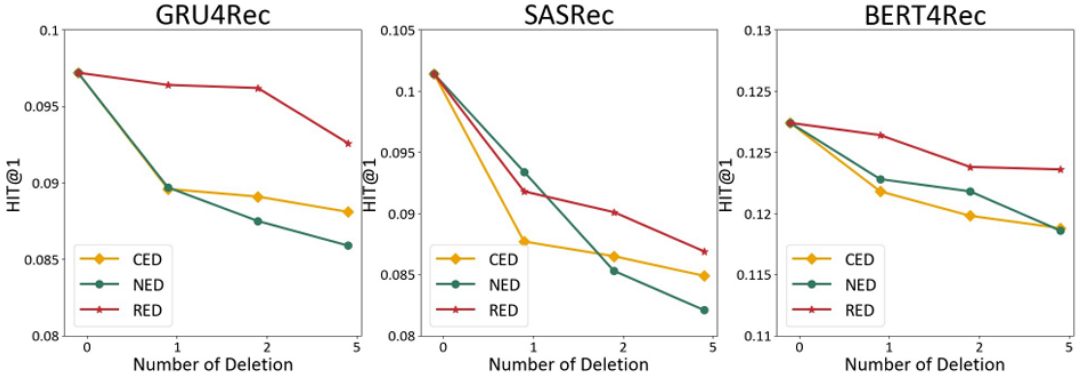

Figure 5: Effect of extra deletion numbers.