Hierarchical Projection Enhanced Multi-behavior Recommendation

Chang Meng*†
mengc21@mails.tsinghua.edu.cn
Shenzhen International Graduate School, Tsinghua University
Shenzhen, China

Hengyu Zhang*†
zhang-hy21@mails.tsinghua.edu.cn
Shenzhen International Graduate School, Tsinghua University
Shenzhen, China

Wei Guo
guowei67@huawei.com
Huawei Singapore Research Center
Singapore

Huifeng Guo
huifeng.guo@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Haotian Liu
liu-ht21@mails.tsinghua.edu.cn
Shenzhen International Graduate School, Tsinghua University
Shenzhen, China

Yingxue Zhang
yingxue.zhang@huawei.com
Huawei Technologies Canada
Montreal, Canada

Hongkun Zheng
zhenghongkun1@huawei.com
Huawei Technologies Co Ltd
Shenzhen, China

Ruiming Tang‡
tangruiming@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Xiu Li‡
li.xiu@sz.tsinghua.edu.cn
Shenzhen International Graduate School, Tsinghua University
Shenzhen, China

Rui Zhang
rayteam@yeah.net
ruizhang.info
Shenzhen, China

ABSTRACT

Various types of user behaviors are recorded in most real-world recommendation scenarios. To fully utilize the multi-behavior infromation, the exploration of multiplex interaction among them is essential. Many multi-task learning based multi-behavior methods are proposd recently to use the multiple types of supervision signals and perform information transfer among them. Despite the great successes, these methods fail to design prediction tasks comprehensively, leading to insufficient utilization of multi-behavior correlative information. Besides, these methods are either based on weighting of expert information extracted from the coupled input or modeling of information transfer between multiple behavior levels through task-specific extractors, which are usually accompanied by negative transfer phenomenon[1]. To address the above problems, we propose a multi-behavior recommendation framework, called Hierarchical Projection Enhanced Multi-behavior Recommendation (HPMR). The key module, Projection-based Transfer Network (PTN), uses the projection mechanism to *"explicitly"* model the correlations of upstream and downstream behaviors, refines the upstream behavior representations, and fully uses the refined representations to enhance the learning of downstream tasks. Offline experiments on public and industrial datasets and online A/B test further verify the effectiveness of HPMR in modeling the associations from upstream to downstream and alleviating the negative transfer. The source code and datasets are available at https://github.com/MC-CV/HPMR.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Multi-behavior, Multi-task Learning, Projection Enhanced

ACM Reference Format:
Chang Meng, Hengyu Zhang, Wei Guo, Huifeng Guo, Haotian Liu, Yingxue Zhang, Hongkun Zheng, Ruiming Tang, Xiu Li, and Rui Zhang. 2023. Hierarchical Projection Enhanced Multi-behavior Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599838

1 INTRODUCTION

Recommender systems are widely used in various online services, such as e-commerce, app store, etc. To provide better services for users and maximize benefits for platforms, advanced recommendation methods are proposed based on users' historical behaviors. In real scenarios, there are multiple kinds of user behaviors. As the example of e-commerce shown in Figure 1 (a), user interacts with items by viewing, carting, purchasing, etc. Therefore, to achieve better performance, multiple behaviors are considered in recommendation models recently. Among the multiple behaviors of users, there is usually a specific behavior (e.g., purchasing) we focus the

*Both authors contributed equally to this research.
†Work done when they were research interns at Huawei Noah's Ark Lab.
‡The corresponding author.
[1]Negative transfer indicates the performance deterioration when harmful information is transferred across different tasks [33]. We define the information that will affect the learning of the target behavior as harmful information.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599838

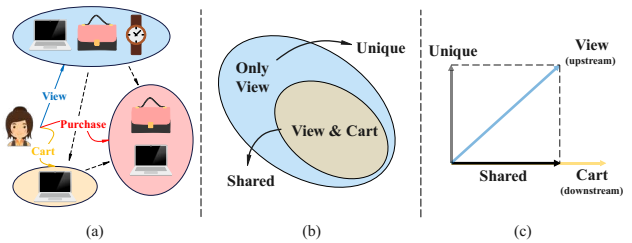


Figure 1: (a) An example of the hierarchical structure of user's multi-behaviors. (b) Correlation of View→Cart (View→Purchase and Cart→Purchase are similar). (c) Illustration of Projection mechanism (View→Cart as an example).

most on, and we define it as the target behavior. In this case, to better predict the user's target behavior, researchers [11, 37, 42] usually probe into other auxiliary behaviors (e.g., viewing, carting, etc.) of users to assist the prediction of the target behavior.

To fully utilize various behaviors, early multi-behavior studies extend matrix factorization [23, 30, 32, 42] or adjust sampling strategies [25, 28] to exploit auxiliary behaviors. However, these shallow methods fail to capture the complicated dependency of multi-behaviors due to the simple architectures.

Recent multi-behavior works typically start from two perspectives [18]: (1) designing advanced neural networks (e.g., deep neural networks and graph neural networks) for multi-behavioral dependencies modeling; (2) devising a Multi-task Learning (MTL) mechanism to fully utilize multiplex interactive information. And these works usually focus on one or both of these perspectives for improvements. For the former perspective, methods utilize attention network [14], transformer [37, 38], and graph neural network [15, 20, 27, 35, 36] to capture the multiplex behavioral dependency of users. To fully use multiple types of interactive information, the latter perspective seeks to design novel MTL structures to provide more supervision signals and knowledge transfer. Recent state-of-the-art multi-behavior methods are mostly based on MTL. For example, some of these methods weight the expert information extracted from coupled representations, and use it to handle different tasks [5, 6, 26, 31]. Besides, there are methods that design MTL modules as transfer-based forms, which are more reasonable in real-world scenarios. In fact, in most cases (e.g., e-commerce scenario), the interactive flow between users and items is often a progressive hierarchical structure from upstream to downstream, expressed as view→cart→purchase or view→purchase (shown in Figure 1 (a)). Naturally, these transfer-based MTL structures establish information transfer among different behaviors and try to correlate the representations of upstream and downstream behaviors through linear mapping [8, 11], meta-learning [39], and the combination form of meta-learning and contrastive learning [36], etc.

Though great progress has been made, existing MTL-based multi-behavior methods still have some limitations when dealing with the characteristic of the hierarchical structure:

- **Incomplete Design of Prediction Tasks.** Existing MTL-based multi-behavior methods do not design the prediction tasks from the perspective of the hierarchical behavior correlations. As shown in Figure 1 (b), supposing that we only consider view and cart for prediction, the correlations between view and cart can be divided into "Only View" and "View & Cart". However, the existing methods ignore "Only View" (as well as "Only Cart" for cart and purchase). In fact, "Only View" and "View & Cart" are complementary. Separately utilizing and predicting the former helps the learning of the latter, thus facilitating both upstream and downstream tasks. In all, it is crucial to design suitable prediction tasks to fully exploit this correlative information.
- **Negative Information Transfer.** Existing MTL-based multi-behavior methods *"implicitly"* model the correlations between different behaviors directly through coupled representations or through information transfer via task-specific extractors, without making full use of behavioral correlations to guide the information interaction among multiple behaviors. This will result in the negative transfer phenomenon that harmful information transferred from the upstream affects the learning of downstream tasks (Proofs are shown in Appendix A.1.1 and A.1.2).

To address these problems, we design a **H**ierarchical **P**rojection Enhanced **M**ulti-behavior **R**ecommendation (HPMR) framework, which contains the Base Model and **P**rojection-based **T**ransfer **N**etwork (PTN). The Base Model mainly contains an *optional* advanced network as the backbone to model higher-order behavioral dependencies. And PTN mainly contains three components, Unique Representation Supervision (URS), Shared Information Transfer (SIT), and Unique Representation Re-projection (URR).

To solve the first problem, inspired by DUMN [4], we use a geometric projection mechanism (shown in Figure 1 (c)) to project upstream behavior representations to downstream to get shared representations (correspond to "View & Cart" in Figure 1 (b)) and unique representations (correspond to "Only View" in Figure 1 (b)). On this basis, we design an auxiliary loss to supervise the learning of the unique representations in different behavior levels, using the label of "Only View/Cart" (URS).

For the second problem, the projection mechanism we used *"explicitly"* models the correlations of upstream and downstream behaviors, which separates upstream behavior information into shared and unique parts. And further, we design Extraction Network to further refine the shared representations of upstream behavior and transfer it downstream (SIT), while avoiding the harmful transfer of unique information that existed in previous transfer-based methods. Besides, we propose a re-projection method, which re-projects unique representations to upstream and then transfers the resulting re-projected representations upstream through the Extraction Network (URR). It makes the process of learning pay more attention to upstream-specific information, thus avoiding upstream-specific information mixing into the shared information transferred downstream. In addition, URS mentioned above fully leverages the label information of unique parts while facilitating the learning of complementary shared parts. All three components in PTN are beneficial for extracting more useful transfer information to enhance downstream while reducing the negative transfer.

In conclusion, our work has the following contributions:

- We highlight the incomplete design of prediction tasks and negative transfer phenomenon in existing MTL methods for multi-behavior recommendation. To handle these problems, we propose a novel framework HPMR to fully utilize auxiliary behavior while reducing negative transfer.
- We design an additional loss in the URS to make the prediction task more complete. To alleviate the negative transfer, we design SIT to explicitly mitigate useless or even harmful information transfer, while using reprojection in URR to alleviate upstream-specific information mixing into the shared information.
- Our model achieves the best performance on all public and industrial datasets compared with several baseline models. Experiments demonstrate the wide compatibility of our proposed HPMR

with different backbones, and further experimental results verify the superiority of PTN as compared with other MTL modules.
- Evaluations on both the offline and online A/B test demonstrate the effectiveness of our proposed HPMR. HPMR has been deployed in an online advertising platform in Huawei and serves millions of daily active users.

2 RELATED WORK

2.1 Multi-behavior Recommendation

Early multi-behavior works extend matrix factorization to multi-behavior data by sharing embeddings, such as CMF [42]. Besides, there are a few works that utilize auxiliary behavioral signals by adjusting sampling strategies like MC-BPR [25]. However, these methods can not probe into the deeper information between the users and the items. Some recent approaches try to capture deeper correlations from multiplex behavioral information in various ways, which can be divided into two perspectives [18]. And recent works typically focus on one or both of these perspectives for improvements. One perspective uses advanced neural networks such as transformer and graph neural networks to model the relationships between different behaviors. For instance, MATN [37] uses transformer to encode the interactions of multiple behaviors. MGNN [41], MBGCN [20], GHCF [6], and MBGMN [39] propose to utilize message propagation on graphs to model high-order multi-behavioral information. Further, CML [36] combines contrastive learning and GNN to dig out higher-order information between nodes, effectively modeling personalized multi-behavior correlations. The other perspective designs a variety of MTL modules with different structures to model the multiplex interactive information. These diverse MTL structures can be divided into two types. One is to extract expert information from the coupled representation and use it directly for different tasks. For example, MGNN [41] and GHCF [6] regard the aggregated representations of different behaviors as coupled input and use aggregated representations to predict each behavior individually. However, these approaches model the correlations between different behaviors directly through coupled representations, which leads to negative transfer phenomenon. The other type uses a transfer-based form to formulate the association between different tasks. For instance, NMTR [11] and EHCF [8] design a linear mapping structure to transfer and aggregate different behavioral representations and then make predictions separately. Besides, MBGMN [39] and CML [36] propose meta-learning and contrastive meta-learning paradigms to distill transferable knowledge across different types of behaviors, after which they transfer it from upstream to downstream. However, these models transfer the extracted knowledge in an *"implicit"* way via task-specific extractors, which will also lead to negative transfer phenomenon. So we propose a projection-based method to *"explicitly"* model the correlations between upstream and downstream, and extract the transfer information according to the correlations.

2.2 Multi-Task Learning for Recommendation

Multi-Task Learning (MTL) is a learning paradigm in machine learning and its aim is to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks. The most common method of multi-task learning is Shared Bottom [5], which uses the coupled input to predict each task individually. In order to deal with the differences between tasks, some studies have applied attention networks to fuse information. For example, MOE [19], MMOE [26] further utilizes different gating networks to obtain different fusion weights in MTL. However, the above methods cannot extract the specific knowledge of different tasks, accompanied by task conflicts and the negative transfer phenomenon (i.e., the performance deterioration when information is transferred across different tasks). To solve this problem, PLE further proposes an extraction paradigm that combines shared and task-specific experts, which adaptively utilizes these experts to deal with task conflicts and alleviate negative transfer phenomenon. However, as all of the existing MTL methods use the coupled representations of all behaviors as input, the negative transfer phenomenon can not be fully resolved due to the gradient issue.

3 PRELIMINARY

3.1 Problem Definition

We use u and v to denote a user and an item, $\mathbf{U}$ and $\mathbf{V}$ denote the user and item sets, respectively. The user-item interactive information of K behavior types can be denoted as a matrix set $\mathcal{M} = \{\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_K\}$, where $\mathbf{M}_k = \left[m_{(k)uv}\right]_{|\mathbf{U}|\times|\mathbf{V}|} \in \{0, 1\}$ indicates whether the user u interacted with the item v under behavior k. Moreover, the user-item interaction data can be regarded as a user-item bipartite graph $\mathcal{G} = (\mathcal{H}, \mathcal{E}, \mathcal{M})$, where $\mathcal{H} = \mathbf{U} \cup \mathbf{V}$, $\mathcal{E} = \cup_{k=1}^{K} \mathcal{E}_k$ is the edge set including all behavior records between users and items. To conveniently illustrate the hierarchical structure of the behavior, we set the value of $k \in \{1, 2, \cdots, K\}$ to denote the order (i.e., 1 and K correspond to the most upstream and the most downstream, respectively). For multi-behavior recommendation, there exists a target behavior (denotes as $\mathbf{M}_K$) to be optimized, which is purchasing (buying) for e-commerce scenarios. Here, the target behavior is the most downstream behavior.

3.2 Unique Interactions Definition

To sufficiently use the correlative information, we separate unique interactions between multiple user behavior pairs (such as viewing without carting and carting without buying) from the original interactive information. In details, we define $\mathcal{M}^{uni} = \{\mathbf{M}_{ij}^{uni} | \mathbf{M}_{ij}^{uni} = \mathbf{M}_i \odot (J - \mathbf{M}_j), i < j, i, j \in \{1, 2, \cdots, K\}\}$, where J is a matrix completely filled with ones, $\odot$ is the element-wise product operator. $\mathcal{M}^{uni}$ is used to fully utilize the interactive information of the unique part of behaviors. Similar to the last part, we define $\mathcal{G}^{uni} = (\mathcal{H}^{uni}, \mathcal{E}^{uni}, \mathcal{M}^{uni})$, where $\mathcal{H}^{uni} = \mathcal{H} = \mathbf{U} \cup \mathbf{V}$, $\mathcal{E}^{uni} = \cup_{k=1}^{K} \mathcal{E}_k^{uni}$.

4 METHOD

We propose a Hierarchical Projection Enhanced Multi-behavior Recommendation (HPMR) framework, which contains the Base Model and the Projection-based Transfer Network. Figure 2 illustrates the technical details of the proposed framework.

4.1 Base Model

As a complete module, Base Model can be directly used to predict different behaviors of users, which consists of Embedding Layer, Encoding Network and Training Objective. Noticed that the Base Model can also be replaced by other models like MBGMN [39], EHCF [8] and GHCF [6], etc. Therefore, we make a compatibility analysis in Section 5.7 to verify the generality of our proposed

HPMR framework with these models as backbones. Here are details of the default Base Model.

4.1.1 Embedding Layer. In order to better model the information of hundreds of thousands of users and items in industrial recommender scenarios, high-dimensional one-hot vectors are often used to represent them. For any given user-item pair (u,i), we have:

$$\mathbf{x}_u = \mathbf{E}_u^T \cdot \mathbf{s},\ \mathbf{y}_v = \mathbf{E}_v^T \cdot \mathbf{t}, \tag{1}$$

where $\mathbf{E}_u \in \mathbb{R}^{|\mathbf{U}|\times d}$ and $\mathbf{E}_v \in \mathbb{R}^{|\mathbf{V}|\times d}$ are the created embedding tables for users and items, $\mathbf{s} \in \mathbb{R}^{|\mathbf{U}|}$ and $\mathbf{t} \in \mathbb{R}^{|\mathbf{V}|}$ denotes the one-hot IDs of user u and item v, $|\mathbf{U}|$ and $|\mathbf{V}|$ are the numbers of users and items, and d is the embedding size.

4.1.2 Encoding Network. Graph convolutional networks are widely used to model higher-order interactions between users and items. Existing works [6, 20, 39, 40] show that the GCN-based methods are extremely effective in mining deeper interactive information between users and items under different behaviors. Thus, to capture the complex information of different behaviors, we use a GCN-based paradigm to model the high-order interactions:

$$\mathbf{x}_u^{k,l} = \underset{v\in N_u^k}{Agg}\left(\frac{\mathbf{y}_v^{k,l-1} \odot \mathbf{r}^{k,l-1}}{\sqrt{|N_u^k||N_v^k|}}\right), \mathbf{r}^{k,l} = \mathbf{W}^l \mathbf{r}^{k,l-1}, \tag{2}$$

where $\odot$ denotes the element-wise product of vectors. N_u^k and N_v^k denote the set of neighbors of u and v under the behavior k, respectively. $k \in \{1, 2, \cdots, K\}$, l denotes the l-th layer. $\mathbf{W}^l$ is a layer specific parameter. $\frac{1}{\sqrt{|N_u^k||N_v^k|}}$ is the symmetric normalization value to adjust the scale of representations. $\mathbf{r}^{k,l}$ is the behavior-specific relational weight. Besides, we have $\mathbf{x}_u^{k,0} = \mathbf{x}_u$. And $\mathbf{r}^{k,0} \in \mathbb{R}^{1\times d}$ is a randomly initialized vector. Meanwhile, we can get $\mathbf{y}_v^{k,l}$ for item v by iterating in a similar way to Equation (2).

We apply the message propagation and aggregation on each particular behavior, thus we can get the representations of different behaviors. Besides, as for the message propagation, we adopt the state-of-the-art GCN models, such as LightGCN [16], LR-GCCF [9], GCN [22] and NGCF [35], for graph information aggregation. Different from GHCF [6] which outputs a weighted sum of representations of different behaviors, we independently output each behavior representation. Thus, for each behavior k, we have:

$$\mathbf{e}_u^k = \sum_{l=0}^{L} \frac{\mathbf{x}_u^{k,l}}{L+1}, \mathbf{e}_v^k = \sum_{l=0}^{L} \frac{\mathbf{y}_v^{k,l}}{L+1}, \mathbf{e}_r^k = \sum_{l=0}^{L} \frac{\mathbf{r}^{k,l}}{L+1}, \tag{3}$$

where L is the number of encoding layers.

4.1.3 Training Objective. In order to learn all parameters more effectively, we follow GHCF [6] and apply the efficient non-sampling learning loss as training objective to optimize our proposed HPMR. The equation of the loss for each behavior k is as follows:

$$\begin{aligned}\tilde{\mathcal{L}}_k^{ori} = &\sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}_u^{k+}}\left(\left(c_v^{k+}-c_v^{k-}\right)(\hat{o}_{uv}^{k,ori})^2 - 2c_v^{k+}\hat{o}_{uv}^{k,ori}\right)\\ &+\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\left(\mathbf{e}_{r,i}^k\mathbf{e}_{r,j}^k\right)\left(\sum_{u\in\mathbf{U}}\mathbf{e}_{u,i}^k\mathbf{e}_{u,j}^k\right)\left(\sum_{v\in\mathbf{V}}c_v^{k-}\mathbf{e}_{v,i}^k\mathbf{e}_{v,j}^k\right)\right),\end{aligned} \tag{4}$$

where $\hat{o}_{uv}^{k,ori} = (\mathbf{e}_u^k)^T \cdot diag\left(\mathbf{e}_r^k\right) \cdot \mathbf{e}_v^k$. c_v^k denotes the weight of entry o_{uv}^k, and it is simplified from c_{uv}^k for efficiency [6]. $\mathbf{V}_u^{k+}$ denotes the interacted items of user u under the behavior k. Detailed explanation of the equation is provided in the Appendix A.3.

4.2 Projection-based Transfer Network

To further model the dependencies among different behaviors, existing methods use MTL modules to fully utilize the multiplex behavioral signals to supervise the process of training. However, these methods fail to design prediction tasks comprehensively, leading to insufficient utilization of multi-behavior correlative information. Besides, these methods are either based on weighted expert information extracted from the coupled representation or modeling information transfer between multiple behavior levels through task-specific extractors, usually accompanied by negative transfer phenomenon. To handle the drawbacks of the existing MTL structures, we propose **P**rojection-based **T**ransfer **N**etwork (PTN), which contains three components (Unique Representation Supervision, Shared Information Transfer and Unique Representation Re-projection). These components work together, explicitly model the correlations of upstream and downstream behaviors and fully utilize the refined representations to enhance the learning of downstream tasks.

First of all, inspired by DUMN [4], we apply a projection mechanism to upstream and downstream behavior representations (as shown in Figure 2). For the sake of simplicity, we present a unified form of user and item:

$$\begin{cases}\mathbf{e}_{sha} = \dfrac{\mathbf{e}_{up}\cdot\mathbf{e}_{down}}{|\mathbf{e}_{down}|}\dfrac{\mathbf{e}_{down}}{|\mathbf{e}_{down}|},\\ \mathbf{e}_{uni} = \mathbf{e}_{up} - \mathbf{e}_{sha},\end{cases} \tag{5}$$

where $(\cdot)$ is the vector inner product operation. $\mathbf{e}_{up}$ and $\mathbf{e}_{down}$ are the representations of the upstream and downstream behaviors, respectively. $\mathbf{e}_{sha}$ contains a mixture information of the upstream and downstream behaviors. $\mathbf{e}_{uni}$, which represents the unique part of the upstream behavior, and $\mathbf{e}_{down}$, which denotes the downstream behavior, are distinctive and should be as orthogonal as possible.

For convenience, we use $\mathbf{e}_{sha}^{k_1k_2}$ to stand for the shared part between behavior k_1 (upstream) and k_2 (downstream). While, $\mathbf{e}_{uni}^{k_1k_2}$ denotes the unique part of behavior k_1 relative to k_2.

4.2.1 Unique Representation Supervision (URS). Existing MTL methods for multi-behavior recommendation ignore the unique part of upstream behavior (correspond to "Only View" shown in Figure 1 (b)). The unique part of upstream behavior contains the semantic information that is complementary to the shared part. Fully using this interactive information can facilitate the extraction of shared information. To do this, we propose Unique Representation Supervision (URS). In particular, we design a unique supervised loss, which uses the unique representation $\mathbf{e}_{uni}^{k_1k_2}$ to predict the label of "Only k_1" (i.e. $\mathbf{M}_{k_1k_2}^{uni}$ shown in Section 3.2).

As we have got the unique representation $\mathbf{e}_{uni}^{k_1k_2}$ which is orthogonal to $\mathbf{e}_{sha}^{k_1k_2}$ and $\mathbf{e}^{k2}$ in the above sections, we design a complementary task which utilizes $\mathbf{e}_{uni}^{k_1k_2}$ to learn the information of the unique part of upstream behavior (k_1). For the output, we have:

$$\hat{o}_{uv}^{k_1k_2,uni} = (\mathbf{e}_{u,uni}^{k_1k_2})^T \cdot diag\left(\mathbf{e}_r^{k_1,uni}\right) \cdot \mathbf{e}_{v,uni}^{k_1k_2}, \tag{6}$$

where $\mathbf{e}_r^{k_1,uni}$ is the initial weight for the complementary task, which is similarly generated as $\mathbf{e}_r^k$ in Section 4.1.2. Here, k_1 represents the upstream behavior relative to k_2.

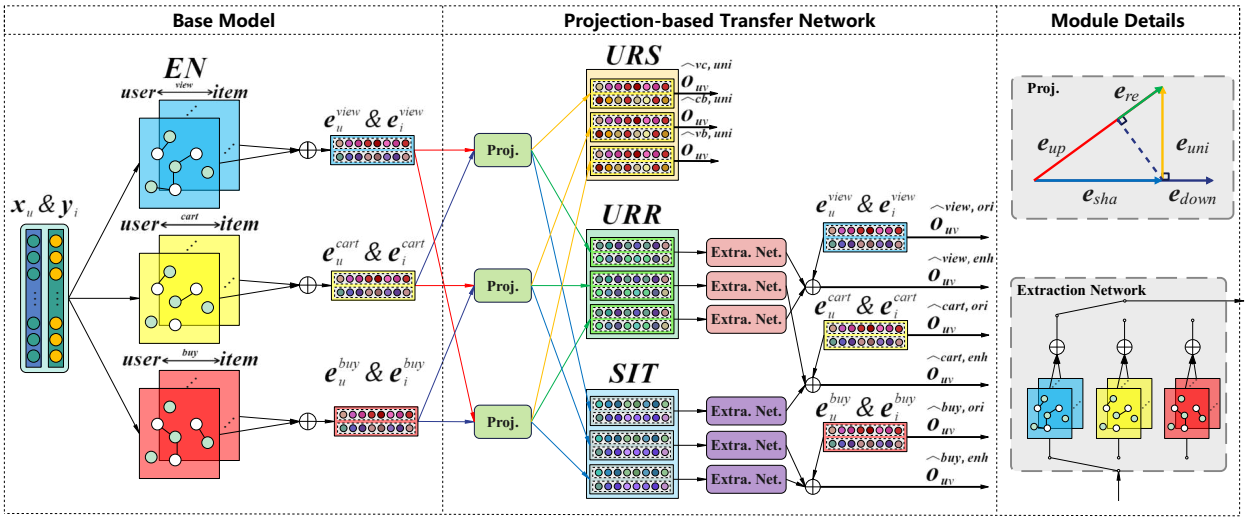


Figure 2: Illustration of the proposed HPMR. Lines of different colors correspond to representations of different colors in the Proj. module (e.g., red lines denote e_{up} and blue lines represent e_{sha}). ($\oplus$) denotes the element-wise addition operation.

Further, we adopt non-sampling learning loss:

$$\tilde{\mathcal{L}}_k^{uni} = \sum_{k_t=k+1}^{K} \left(\sum_{u \in \mathbf{U}} \sum_{v \in \mathbf{V}_{u,uni}^{kk_t+}} \left(\left(c_v^{k+} - c_v^{k-} \right) (\hat{o}_{uv}^{kk_t,uni})^2 - 2 c_v^{k+} \hat{o}_{uv}^{kk_t,uni} \right) \right.$$
$$\left. + \sum_{i=1}^{d} \sum_{j=1}^{d} \left(\left(\mathbf{e}_{r,i}^{k,uni} \mathbf{e}_{r,j}^{k,uni} \right) \left(\sum_{u \in \mathbf{U}} \mathbf{e}_{u,i}^{kk_t} \mathbf{e}_{u,j}^{kk_t} \right) \left(\sum_{v \in \mathbf{V}_{u,uni}^{kk_t+}} c_v^{k-} \mathbf{e}_{v,i}^{kk_t} \mathbf{e}_{v,j}^{kk_t} \right) \right) \right), \tag{7}$$

where k_t is the downstream relative to k (i.e., $k < k_t \leq K$). $\mathbf{V}_{u,uni}^{kk_t+}$ denotes the interacted items of user u recorded in the matrix $\mathbf{M}_{kk_t}^{uni}$. For simplicity, we set $\mathbf{e}_u^{kk_t} = \mathbf{e}_{u,uni}^{kk_t}$, $\mathbf{e}_v^{kk_t} = \mathbf{e}_{v,uni}^{kk_t}$ in this equation.

4.2.2 Shared Information Transfer (SIT). After projection, we have got the shared part ($\mathbf{e}_{sha}$) and the unique part ($\mathbf{e}_{uni}$) between the upstream and downstream. Then we only transfer the shared representation downstream avoiding harmful transfer of unique part. Inspired by the gate mechanism of MTL [26, 31], we designed an alternative Extraction Network based on GNN as a task-specific extractor to further refine $\mathbf{e}_{sha}$ and transfer it downstream. Similar to Section 4.1.2, we have:

$$\mathbf{e}_{u,sha}^{k_1k_2,l} = \underset{v \in N_u^{k_2}}{Agg} \left(\frac{\mathbf{e}_{v,sha}^{k_1k_2,l-1} \odot \mathbf{e}_r^{k_2,l-1}}{\sqrt{|N_u^{k_2}||N_v^{k_2}|}} \right), \mathbf{e}_{r,sha}^{k_2,l} = \mathbf{W}^l \mathbf{e}_{r,sha}^{k_2,l-1}, \tag{8}$$

where $\odot$ denotes the element-wise product of vectors. $\mathbf{e}_{v,sha}^{k_1k_2,0} = \frac{\mathbf{e}_v^{k_1} \cdot \mathbf{e}_v^{k_2}}{|\mathbf{e}_v^{k_2}|} \cdot \frac{\mathbf{e}_v^{k_2}}{|\mathbf{e}_v^{k_2}|}$ and $\mathbf{e}_{r,sha}^{k_2,0} = \mathbf{e}_r^{k_2}$. $\mathbf{W}^l$ is the layer specific parameter shared with the layer of Encoding Network. Meanwhile, we can get $\mathbf{e}_{v,sha}^{k_1k_2,l}$ for item v by iterating in a similar way to Equation (8).

Thus, for the output, we have:

$$\mathbf{g}_{u,sha}^{k_1k_2} = \sum_{l=0}^{L_{tr}} \frac{\mathbf{e}_{u,sha}^{k_1k_2,l}}{L_{tr}+1}, \mathbf{g}_{v,sha}^{k_1k_2} = \sum_{l=0}^{L_{tr}} \frac{\mathbf{e}_{v,sha}^{k_1k_2,l}}{L_{tr}+1}, \tag{9}$$

where L_{tr} is the number of transfer layers.

4.2.3 Unique Representation Re-projection (URR). Since the transferred shared information may be mixed with upstream-specific information, we design Unique Representation Re-projection (URR). In particular, we propose a re-projection mechanism, which enhances the unique part of upstream without changing the correlations between upstream and downstream behavioral representations (Illustrated in Appendix A.2). It makes the process of learning pay more attention to upstream-specific information, thus avoiding upstream-specific information mixing into the shared information transferred downstream. As shown in Figure 2, we re-project the $\mathbf{e}_{uni}$ to $\mathbf{e}_{up}$, and get the re-projection representation which contains a mixture of information of the unique part of the upstream and the whole upstream. Similar to Section 4.2.2, we further pass the $\mathbf{e}_{re}$ to Extraction Network. In detail, we have:

$$\mathbf{e}_{u,re}^{k_1k_2,l} = \underset{v \in N_u^{k_1}}{Agg} \left(\frac{\mathbf{e}_{v,re}^{k_1k_2,l-1} \odot \mathbf{e}_r^{k_1,l-1}}{\sqrt{|N_u^{k_1}||N_v^{k_1}|}} \right), \mathbf{e}_{r,re}^{k_1,l} = \mathbf{W}^l \mathbf{e}_{r,re}^{k_1,l-1}, \tag{10}$$

where $\odot$ denotes the element-wise product of vectors. $\mathbf{e}_{v,re}^{k_1k_2,0} = \frac{\mathbf{e}_{v,uni}^{k_1k_2} \cdot \mathbf{e}_v^{k_1}}{|\mathbf{e}_v^{k_1}|} \frac{\mathbf{e}_v^{k_1}}{|\mathbf{e}_v^{k_1}|}$ and $\mathbf{e}_{r,re}^{k_1,0} = \mathbf{e}_r^{k_1}$. $\mathbf{W}^l$ is the layer specific parameter shared with the layer of Encoding Network. Meanwhile, we can get $\mathbf{e}_{v,re}^{k_1k_2,l}$ for item v by iterating in a similar way to Equation (10).

Thus, for the output, we have:

$$\mathbf{g}_{u,re}^{k_1k_2} = \sum_{l=0}^{L_{tr}} \frac{\mathbf{e}_{u,re}^{k_1k_2,l}}{L_{tr}+1}, \mathbf{g}_{v,re}^{k_1k_2} = \sum_{l=0}^{L_{tr}} \frac{\mathbf{e}_{v,re}^{k_1k_2,l}}{L_{tr}+1}, \tag{11}$$

where L_{tr} is the number of transfer layers (same as Section 4.2.2).

4.3 Joint Optimization

In previous parts, we have obtained the shared information transfer output $\mathbf{g}_{u,sha}^{k_1k_2}$, $\mathbf{g}_{v,sha}^{k_1k_2}$ and the re-projection output $\mathbf{g}_{u,re}^{k_1k_2}$ and $\mathbf{g}_{v,re}^{k_1k_2}$. In the process of information propagation and aggregation, we have gathered information from the item side to the user side, so here we've only enhanced the user side. In detail, we have:

$$\begin{cases} \mathbf{g}_u^k = \mathbf{e}_u^k + \alpha * \sum_{k_i=1}^{k-1} \mathbf{g}_{u,sha}^{k_ik} + \beta * \sum_{k_j=k+1}^{K} \mathbf{g}_{u,re}^{kk_j}, \\ \hat{o}_{uv}^{k,enh} = (\mathbf{g}_u^k)^T \cdot diag\left(\mathbf{e}_r^k\right) \cdot \mathbf{e}_v^k, \end{cases} \tag{12}$$

Table 1: Statistics of evaluation datasets.

Dataset	#Users	#Items	#View	#Add-to-cart	#Purchase
Beibei	21,716	7,977	2,412,586	642,622	304,576
Taobao	48,749	39,493	1,548,126	193,747	259,747

where $k \in \{1, 2, \cdots, K\}$ and we set that $\sum_a^b = 0$ when $a > b$. $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are coefficients representing the weights of shared information and unique information, respectively. Based on the Equation (4), we replace the e_u^k to $\mathbf{g}_u^k$, thus getting the representation enhanced loss $\tilde{\mathcal{L}}_k^{enh}$.

Last but not least, in order to get a better model for each behavioral task [3], following most multi-behavior tasks [6, 8, 11, 20], we apply a MTL form to better learn parameters from the data of different behaviors:

$$\mathcal{L}(\Theta) = \sum_{k=1}^{K} \lambda_k (\gamma_1 \tilde{\mathcal{L}}_k^{ori} + \gamma_2 \tilde{\mathcal{L}}_k^{enh} + \gamma_3 \tilde{\mathcal{L}}_k^{uni}) + \mu \|\Theta\|_2^2, \tag{13}$$

where λ_k is the weight to control the influence of the k-th behavior on the joint training. Moreover, we follow GHCF and enforce that $\sum_{k=1}^{K} \lambda_k = 1$. $\gamma_1, \gamma_2, \gamma_3 \geq 0$ are coefficients of the three losses. L_2 regularization parameterized by μ on Θ is conducted to prevent overfitting.

4.4 Complexity Analysis

4.4.1 Time Complexity. We spend $O(L\,|\mathcal{E}|\,d)$ for message propagation in the Encoding Network, where L denotes the number of encoding layers, $|\mathcal{E}|$ is the number of edges on $\mathcal{G}$ and d is the embedding size. After that, the time spent on PTN mainly comes from the Extraction Network, which spends $O(L_{tr}\,|\mathcal{E}|\,d)$. L_{tr} is the number of layers of the Extraction Network. Besides, the time complexity of the calculation of loss is illustrated in Appendix A.3. In all, the overall time complexity of HPMR mainly comes from the GNN part, which is comparable to other GNN-based methods.

4.4.2 Space Complexity. Parameters that our proposed HPMR needs to learn mainly come from the embeddings of users and items, which costs $O((|\mathbf{U}| + |\mathbf{V}|)d)$. Besides, as all dense sub-graphs in $\mathcal{G}$ and $\mathcal{G}^{uni}$ are transformed into sparse behavior-specific graphs, no extra memory space is needed to store these graphs. In all, HPMR has limited additional parameters except for the embeddings of the users and items.

5 EXPERIMENTS

5.1 Datasets

To validate the effectiveness of our proposed HPMR model, we conduct experiments on two publicly available datasets (Beibei and Taobao)[2] with multiple behaviors such as view, add-to-cart, and purchase. For a fair comparison, the pre-processing and splitting methods of two datasets are the same as those used in GHCF [6]. The details of datasets are listed in Table 1.

5.2 Evaluation Metrics

For all experiments, we evaluate our proposed HPMR and baseline models in terms of the top-k recommended items with two metrics, i.e., the Hit Ratio (*HR@k*) and the Normalized Discounted Cumulative Gain (*NDCG@k*). In particular, we set $k = 10$.

[2]https://github.com/chenchongthu/GHCF

5.3 Baseline Models

To sufficiently demonstrate the validity of HPMR, we compare it with a variety of baseline models, which can be divided into three groups: **(1) Single-behavior methods:** BPR [29], NCF [17], ENMF [7] and LightGCN[3] [16], **(2) Multi-behavior methods without MTL:** CMF [42], MC-BPR [25], MBGCN[4] [20] and MATN[5] [37], **(3) Multi-behavior methods with MTL:** NMTR [11], CML[6] [36], MBGMN[7] [39], LightGCN$_M$, EHCF [8] and GHCF[2] [6]. Among them, MATN is a transformer-based model, and CML is a contrastive learning-based model. LightGCN, MBGCN, LightGCN$_M$, MBGMN, CML, and GHCF are GNN-based models. As LightGCN is originally designed for single behavior prediction, we apply multi-behavior and non-sampling learning to it and get LightGCN$_M$.

5.4 Implementation Details

Our proposed HPMR is implemented in TensorFlow [2]. For a fair comparison, following GHCF [6], we set the embedding size and batch size to 64 and 256, respectively. We initialize the parameters using Xavier [12]. The parameters are optimized by Adam [21], while the learning rate is set to 10^{-3}. We search the number of encoding layers and transfer layers in {0,1,2,3,4} for user-item bipartite graph. Besides, we tune the coefficients of the three types of loss in {0,1,2,3,4,5}. The dropout ratio is set to 0.8 and 0.9 for Beibei and Taobao, respectively. Other parameters like the negative weight are the same as GHCF. All experiments are run 5 times and average results are reported. For comprehensiveness and fairness, we present the results of model performance comparison under the two mainstream settings which widely used by existing works. In addition, we conduct hyper-parameter analysis experiments (shown in Appendix A.4).

Table 2: The overall performance comparison. Boldface denotes the highest score and underline indicates the results of the best baselines. ⋆ represents significance level p-value < 0.05 of comparing HPMR with the best baseline.

Dataset	Beibei		Taobao	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10
BPR	0.0437	0.0213	0.0376	0.0227
NCF	0.0441	0.0225	0.0391	0.0233
ENMF	0.0464	0.0247	0.0398	0.0244
LightGCN	0.0451	0.0232	0.0415	0.0237
CMF	0.0482	0.0251	0.0483	0.0252
MC-BPR	0.0504	0.0254	0.0547	0.0263
MBGCN	0.0483	0.0245	0.0400	0.0227
MATN	0.0547	0.0281	0.0417	0.0223
NMTR	0.0524	0.0285	0.0585	0.0278
CML	0.0573	0.0281	0.0623	0.0336
MBGMN	0.1075	0.0536	0.0437	0.0252
LightGCN$_M$	0.1788	0.0935	0.0652	0.0377
EHCF	0.1523	0.0817	0.0717	0.0403
GHCF	<u>0.1922</u>	<u>0.1012</u>	<u>0.0807</u>	<u>0.0442</u>
HPMR	**0.2375⋆**	**0.1352⋆**	**0.0948⋆**	**0.0586⋆**
Rel Impr.	23.57%	33.60%	17.47%	32.58%

[3]https://github.com/kuandeng/LightGCN
[4]https://github.com/tsinghua-fib-lab/MBGCN
[5]https://github.com/akaxlh/MATN
[6]https://github.com/weiwei1206/CML
[7]https://github.com/akaxlh/MB-GMN

5.5 Overall Performance Evaluation

5.5.1 Effectiveness comparison under the setting of full negative samples. Table 2 shows the performance of different methods on Beibei and Taobao datasets with respect to HR@10 and NDCG@10. We have the following findings:

The effectiveness of HPMR. As shown in the table, HPMR achieves the best performances at all datasets. Specifically, HPMR improves the best baselines by **23.57%** and **17.47%** in terms of HR (**33.60%**, and **32.58%** in terms of NDCG) on Beibei and Taobao datasets, respectively. The great improvements over baselines verify the effectiveness of HPMR for multi-behavior recommendation.

Both MTL and Multi-behavior based methods improve the performance. Although different baselines have different learning objectives and network architectures, we can find that the multi-behavior methods with MTL have a consistent trend and perform much better than those multi-behavior approaches without MTL. For example, by fully utilizing the multiplex interactive information, EHCF and GHCF outperform MBGCN and MATN in all datasets and metrics in the multi-behavior settings. Besides, multi-behavior models MATN and MC-BPR achieve much better performance than single-behavior models ENMF and LightGCN, which further verifies the effectiveness of using the auxiliary behavior information for learning.

HPMR consistently outperforms MTL-based multi-behavior baseline models. Our proposed HPMR performs better than the multi-behavior model LightGCN$_M$, EHCF and GHCF which are based on MTL. Through the design of projection, Extraction Network, and complementary task, we fully refine the upstream behavior representation and transfer more useful information to the downstream. Besides, HPMR explicitly models and exploits the dynamic correlations between upstream and downstream behavior information. While the existing multi-behavior models with MTL do not model or can only implicitly model the task-specific correlations between different behaviors, thus leading to the negative transfer phenomenon between behaviors.

Table 3: The overall performance comparison under the setting of 99 negative samples. Boldface denotes the highest score and underline indicates the results of the best baselines. $\star$ represents significance level p-value < 0.05 of comparing HPMR with the best baseline.

Dataset	Beibei		Taobao	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10
CML	0.6993	0.4250	0.7078	0.4645
MBGMN	0.7295	0.4410	0.6110	0.3853
LightGCN$_M$	0.7478	0.5489	0.7142	0.5022
EHCF	0.7438	0.5449	<u>0.7477</u>	<u>0.5368</u>
GHCF	<u>0.7539</u>	<u>0.5549</u>	0.7387	0.5208
HPMR	**0.7718***	**0.5656***	**0.7817***	**0.5574***

5.5.2 Effectiveness comparison under the setting of 99 negative samples. Many multi-behavior methods use another setting for effectiveness comparisons, such as MBGMN and CML. For a comprehensive comparison, following the setting of them, we compare our HPMR with them and some advanced methods. Specifically, we take the last item in the test data that interacts with the behavior to be predicted as a positive example, and the 99 items randomly selected that users do not interact with as a negative example. As shown in Table 3, we can find that our HPMR still performs best under this setting. The results show that our model has good robustness under different experiment settings.

5.6 Ablation Study

Table 4: Performance of different HPMR variants. $\star$ represents significance level p-value < 0.05 of comparing HPMR with other variants.

Dataset	Beibei		Taobao	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10
HPMR w/o PTN	0.1808	0.0935	0.0636	0.0368
HPMR w/o URS	0.2059	0.1081	0.0803	0.0467
HPMR w/o SIT	0.2008	0.1112	0.0670	0.0402
HPMR w/o URR	0.2277	0.1277	0.0916	0.0560
HPMR	**0.2375***	**0.1352***	**0.0948***	**0.0586***

5.6.1 The ablation of modules in Projection-based Transfer Network. The main part of HPMR is Projection-based Transfer Network (PTN), which contains Unique Representation Supervision (URS), Shared Information Transfer (SIT) and Unique Representation Re-projection (URR). In order to deeply analyze the role of each module, we designed several variants based on HPMR. **HPMR w/o PTN**: We remove the whole PTN, and use the Base Model to train and predict. **HPMR w/o URS**: We remove the Unique Representation Supervision module in PTN. **HPMR w/o SIT**: We remove the Shared Information Transfer module in PTN. **HPMR w/o URR**: We remove the Unique Representation Re-projection module in PTN.

The performance of HPMR and its variants are shown in Table 4, and we have the following conclusions:

- Compared HPMR with the last three variants in the table, we can find that after removing any module of PTN, the model's performance will significantly decrease on all datasets. This fully demonstrates the effectiveness and rationality of our designed module.
- After removing the whole PTN module, the model performance decreases significantly. More specifically, the performance in terms of NDCG on Beibei and Taobao decreased by **30.84%** and **37.20%**, respectively. These results show that PTN as a well-designed MTL module can make the model perform better, and its rationality and effectiveness are self-evident.

Table 5: Impact of MTL Module. Boldface denotes the highest score and underline indicates the results of the best variants.

Dataset	Beibei		Taobao	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10
Base+Shared Bottom	0.1622	0.0826	0.0636	0.0371
Base+MMOE	0.1717	0.0875	0.0627	0.0362
Base+PLE	0.1590	0.0800	0.0577	0.0329
Base+Linear Trans.	0.1801	0.0936	0.0195	0.0103
Base+Target Atten. Trans.	0.1875	0.0990	0.0611	0.0352
Base+Vanilla Atten. Trans.	0.1889	0.0995	0.0610	0.0351
Base+GNN Trans.	0.1835	0.0968	0.0645	0.0371
Base+Proj. Trans.	<u>0.1971</u>	<u>0.1097</u>	<u>0.0682</u>	<u>0.0411</u>
Base+PTN	**0.2375**	**0.1352**	**0.0948**	**0.0586**

5.6.2 Performance comparison between Projection-based Transfer Network and other MTL modules. In order to further prove the superiority of our proposed Projection-based Transfer Network, we replace it with some other MTL modules. As shown in the Table 5, the first three are classic multi-task learning methods, i.e. Shared Bottom [5], MMOE [26] and PLE [31]. While other MTL variants are transfer-based modules which are illustrated as follows: **Linear Trans.**: Similar to NMTR [11], we use linear transformation weights to transfer upstream information. **Target Atten. Trans.**: We apply the Target Attention [44] to aggregate and transfer upstream representations by the guidance of downstream representations. **Vanilla Atten. Trans.**: We conduct the Vanilla Attention [43] to model behavioral interactions between upstream representations and downstream representations. **GNN Trans.**: We transfer the upstream representations through a GNN-based Network to downstream. **Proj. Trans.**: We directly remove the Extraction Network of PTN to get a weakened PTN. As the results are shown in the table, we can find that:

- PTN performs the best among the transfer-based variants and the classic MTL methods on Beibei and Taobao. This fully reflects the effectiveness and superiority of our designed PTN. The details of the theoretical analysis are shown in Appendix A.1.
- The first three transfer-based MTL modules outperform the classic MTL methods on Beibei, while the performance on Taobao is not as good as theirs. A probable reason is that they just use the weights to transfer and aggregate information in representations, making it harder to capture effective information upstream from more complex datasets. While the GNN Trans. utilizes the graph neural network to further extract more complicated information and transfer it to downstream, so that deeper information can be fully leveraged. Besides, Proj. Trans. outperforms all the other variants on the two datasets, which demonstrates the effectiveness of modeling associations between upstream and downstream using explicit projections.

5.7 Compatibility Analysis

Table 6: Compatibility performance of HPMR with different models as backbones ("X+PTN" means using X to replace the backbone in Base Model).

Dataset	Beibei		Taobao	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10
MBGMN	0.1075	0.0536	0.0437	0.0252
MBGMN+PTN	**0.1321**	**0.0691**	**0.0464**	**0.0279**
LightGCN$_M$	0.1788	0.0935	0.0652	0.0377
LightGCN$_M$+PTN	**0.2185**	**0.1220**	**0.0876**	**0.0528**
EHCF	0.1523	0.0817	0.0717	0.0403
EHCF+PTN	**0.2030**	**0.1051**	**0.0840**	**0.0502**
GHCF	0.1922	0.1012	0.0807	0.0442
GHCF+PTN	**0.1973**	**0.1101**	**0.0845**	**0.0539**

Our proposed HPMR is not only a specific model, but also a general framework that can be applied to most existing multi-behavior methods. To ascertain our claim, we replace the backbone in Base Model with some other representative multi-behavior methods, like MBGMN, LightGCN$_M$, EHCF and GHCF, and evaluate the resulting models by comparing them with the original backbones.

We can see that the PTN module in HPMR improves the performance of all backbone models. Compared with MBGMN and GHCF, the improvement of LightGCN$_M$ and EHCF are more significant, which may be because these two models have more superficial structures and more robust compatibility. While MBGMN and GHCF are more complicated, which aggregate and transfer more noise in the part of the encoder, leading to the sub-optimal learning of each behavior representation. In summary, the results fully demonstrate the wide compatibility and broad generality of our proposed HPMR with different backbones, in which the PTN can provide significant performance improvements to the backbone model.

6 INDUSTRIAL APPLICATION

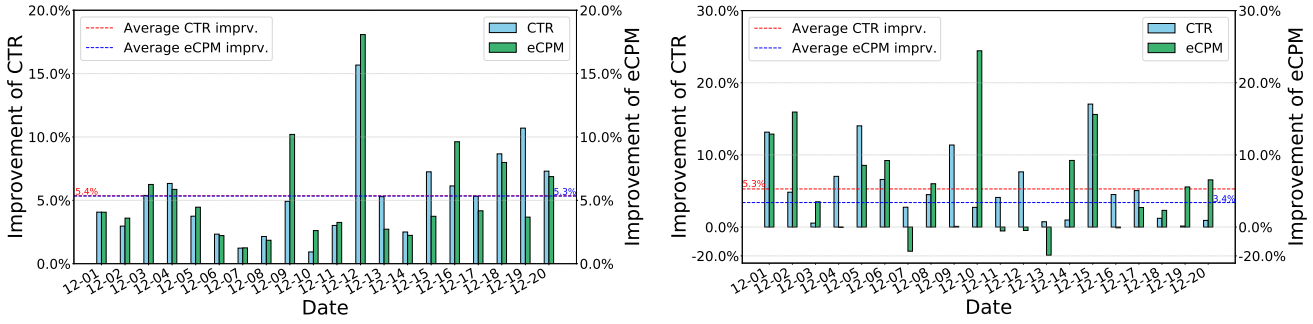
Figure 3: The illustration of the application scenarios.

Our proposed HPMR can not only be used as an auxiliary component to serve the recall stage, but can also be applied to the ranking stage by plugging the PTN module into the CTR model. Since different companies use different CTR models (such as DeepFM [13] in Huawei and DCN [34] in Google), our model has good compatibility and can be widely used in industrial applications. We test the effectiveness of HPMR framework with offline and online evaluation on two real industrial scenarios: **Scenario 1** and **Scenario 2** of the Huawei online advertising platform[8], which has various scenarios, such as Search Aid, Browser, Video Patch, Video Feeds, Video AppStore and Reward Ads, as depicted in Figure 3. These scenarios have diverse ad presentation styles, such as single bar lists (e.g., Video AppStore) and splash ads (e.g., Browser). In addition, the content of ads is also rich and diverse, including products, applications, etc. Meanwhile, the same ads can be displayed in different scenarios with different materials (such as pictures and video profiles). Millions of daily active users are involved in the platform and tens of millions of log events are generated every day.

Table 7: Performance comparison on industrial datasets.

Dataset	Scenario 1		Scenario 2	
Metrics	AUC	Imprv.	AUC	Imprv.
Baseline	0.8088	-	0.7761	-
Ours	**0.8117**	**3.59‰**	**0.7775**	**1.80‰**

[8]https://developer.huawei.com/consumer/en/huawei-ads



(a) Scenario 1 (b) Scenario 2

Figure 4: Online A/B test experimental results of CTR and eCPM.

Table 8: Online A/B test results.

Scenario	Scenario 1			Scenario 2		
Metrics	CTR	eCPM	Inference	CTR	eCPM	Inference
Imprv.	**+5.4%**	**+5.3%**	**+17.7%**	**+5.3%**	**+3.4%**	**+16.8%**

6.1 Offline Evaluation

6.1.1 Datasets. We collect and sample two datasets from these two scenarios for offline evaluation. Concretely, 7 consecutive days of user behavior records are used for training, and the next 2 days for validation and testing. The two datasets have 241 million and 35 million samples, and the corresponding positive ratios are 3.36% and 0.75%, respectively. Compared with the above used two publicly available datasets, these two real-world industrial datasets contain rich side features and contextual information, including user features (e.g., age, download history, etc.), item features (e.g., content, category, etc.) and context features (e.g., site, slot, etc.).

6.1.2 Baseline and Evaluation Metric. The compared baseline is a highly-optimized CTR prediction model. We use **AUC** to evaluate the performance, which is commonly used in real-world scenarios. And it has been claimed in many existing works [10, 13, 24] that a slightly higher AUC (↑) at ***0.001-level*** is regarded significant for CTR prediction tasks.

6.1.3 Performance Comparison. The effectiveness comparison is presented in Table 7. Our method outperforms the baseline model by **3.59‰** and **1.80‰** in terms of AUC on the two datasets. This proves that our method can make better use of the rich and diverse historical behavior information of users for large-scale industrial recommendations.

6.2 Online A/B Test

To further evaluate the online performance of our model in real product environment, we randomly select and serve 5% of users with our model and another 5% of users with baseline model, then deploy them on these two scenarios from 2022-12-01 to 2022-12-20 with 20 days online test.

6.2.1 System Description. The online system consists of three modules: candidate generation, pre-ranking and ranking. Hundreds of thousands of ads are first selected through the candidate generation module to obtain several hundred ads from the entire candidate set for each user. Then, user profiles and the corresponding item features are fed to the pre-ranking module to calculate the preference of the user to specific items, and top-k items with the highest scores are selected and delivered to the next module for more accurate prediction. Finally, the ranking module produces the output list based on the predicted CTR scores and some business principles.

6.2.2 Online Experimental Results. We leverage the commonly used metrics: Click Through Rate (**CTR**), effective Cost Per Mille (**eCPM**), and Latency to evaluate the online performance. Figure 4 shows the results of our model compared to the baseline model on the two scenarios during the test period. Both models are trained with the same dataset and deployed on the same type of cluster servers. We can find that our model achieves average improvements of **5.4%** (**5.3%**) and **5.3%** (**3.4%**) with respect to CTR (eCPM) on the two scenarios respectively, which is a significant improvement and verifies the effectiveness of our proposed method. Besides, as shown in Table 8, the latency of our model is comparable with the baseline model, making our model acceptable in real-world recommendation scenarios. Our model is now deployed online with full users in these scenarios.

7 CONCLUSIONS

In this paper, we propose the HPMR for multi-behavior recommendations. To solve the two problems of existing MTL methods, we design PTN which firstly uses the projection mechanism to explicitly model the correlations of upstream and downstream behaviors, thus separating upstream behavior into shared and unique parts. On this basis, our method fully exploits the unique interactive information of upstream behavior by designing an auxiliary loss, which facilitates the learning of complementary shared parts. Moreover, our method only transfers refined shared representations of upstream behavior downstream, avoiding the negative transfer caused by unique parts. Meanwhile, we propose a re-projection method, making the process of learning pay more attention to upstream-specific information, thus avoiding upstream-specific information mixing into the shared information transferred downstream. Finally, we conduct comprehensive experiments on various datasets, the evaluation on both offline experiments and online A/B test show the high effectiveness and broad generality of our proposed HPMR. HPMR has also been deployed in an online advertising platform in Huawei and serves millions of daily active users. Further analysis verifies the great superiority of PTN with less negative transfer in comparison with other MTL modules.

ACKNOWLEDGMENTS

This work was partly supported by the Science and Technology Innovation 2030-Key Project (Grant No. 2021ZD0201404), Key Technology Projects in Shenzhen (Grant No. JSGG20220831110203007) and Aminer· ShenZhen·ScientificSuperBrain. And we thank MindSpore [1] for the partial support of this work, which is a new deep learning computing framework.

REFERENCES

[1] 2020. MindSpore. https://www.mindspore.cn
[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 265–283.
[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2006. Multi-task feature learning. *Advances in neural information processing systems* 19 (2006).
[4] Zhi Bian, Shaojun Zhou, Hao Fu, Qihong Yang, Zhenqi Sun, Junjie Tang, Guiquan Liu, Kaikui Liu, and Xiaolong Li. 2021. Denoising user-aware memory network for recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 400–410.
[5] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[6] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021. Graph Heterogeneous Multi-Relational Recommendation. In *AAAI*, Vol. 35. 3958–3966.
[7] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient neural matrix factorization without sampling for recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.
[8] Chong Chen, Min Zhang, Yongfeng Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient heterogeneous collaborative filtering without negative sampling for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 19–26.
[9] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*, Vol. 34. 27–34.
[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.
[11] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *ICDE*. IEEE, 1554–1557.
[12] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[13] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).
[14] Long Guo, Lifeng Hua, Rongfei Jia, Binqiang Zhao, Xiaobo Wang, and Bin Cui. 2019. Buying or browsing?: Predicting real-time purchasing intent using attention-based deep network with multiple behavior. In *SIGKDD*. 1984–1992.
[15] Wei Guo, Chang Meng, Enming Yuan, Zhicheng He, Huifeng Guo, Yingxue Zhang, Bo Chen, Yaochen Hu, Ruiming Tang, Xiu Li, et al. 2023. Compressed Interaction Graph based Framework for Multi-behavior Recommendation. In *Proceedings of the ACM Web Conference 2023*. 960–970.
[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).
[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[18] Chao Huang. 2021. Recent Advances in Heterogeneous Relation Learning for Recommendation. *arXiv preprint arXiv:2110.03455* (2021).
[19] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.
[20] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *SIGIR*.
[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[22] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[23] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 173–182.
[24] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*. 1754–1763.
[25] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 361–364.
[26] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *SIGKDD*. 1930–1939.
[27] Chang Meng, Ziqi Zhao, Wei Guo, Yingxue Zhang, Haolun Wu, Chen Gao, Dong Li, Xiu Li, and Ruiming Tang. 2022. Coarse-to-Fine Knowledge-Enhanced Multi-Interest Learning Framework for Multi-Behavior Recommendation. *arXiv preprint arXiv:2208.01849* (2022).
[28] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.
[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
[30] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 650–658.
[31] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *RecSys*. 269–278.
[32] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. 2016. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 283–292.
[33] Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 242–264.
[34] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.
[35] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.
[36] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive meta learning with behavior multiplicity for recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1120–1128.
[37] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Bo Zhang, and Liefeng Bo. 2020. Multiplex behavioral relation learning for recommendation via memory augmented transformer network. In *SIGIR*. 2397–2406.
[38] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4486–4493.
[39] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. 2021. Graph meta network for multi-behavior recommendation. In *SIGIR*. 757–766.
[40] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*. 793–803.
[41] Weifeng Zhang, Jingwen Mao, Yi Cao, and Congfu Xu. 2020. Multiplex Graph Neural Networks for Multi-behavior Recommendation. In *CIKM*. 2313–2316.
[42] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. 2015. Improving user topic interest profiles by behavior factorization. In *Proceedings of the 24th International Conference on World Wide Web*. 1406–1416.
[43] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. Atrank: An attention-based user behavior modeling framework for recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[44] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.

A APPENDIX

A.1 Superiority of the Proposed PTN

A.1.1 The Gradient Issue in Classical MTL. As the classical MTL methods directly couple the representations of different behaviors together with different weights, we have:

$$\mathbf{e}_u^* = \sum_{k=1}^{K} \lambda_k \mathbf{e}_u^k, \mathbf{e}_v^* = \sum_{k=1}^{K} \lambda_k \mathbf{e}_v^k,$$

where K is the number of behaviors, λ_k is the weight of k-th behavior. Taking $(\mathbf{e}_u^*, \mathbf{e}_v^*)$ as input for MTL, the loss function can be formulated as:

$$\mathcal{L}_{uv} = \sum_{k=1}^{K} L(f_k(\mathbf{e}_u^*, \mathbf{e}_v^*) - o_{uv}^k),$$

where $\hat{o}_{uv}^k$ denotes the predictive probability that user u will interact with item v under the k-th behavior, o_{uv}^k is the true label, $L(\cdot)$ is the loss function, and $f_k(\cdot)$ is the predictive function in MTL models. Then we have:

$$\frac{\partial \mathcal{L}_{uv}}{\partial(\mathbf{e}_u^* \circ \mathbf{e}_v^*)} = \sum_{k=1}^{K} \frac{\partial f_k(\mathbf{e}_u^*, \mathbf{e}_v^*)}{\partial(\mathbf{e}_u^* \circ \mathbf{e}_v^*)} * L^{'}(f_k(\mathbf{e}_u^*, \mathbf{e}_v^*) - o_{uv}^k) = \sum_{k=1}^{K} a_{uv}^k \mathbf{r}^k,$$

where $(\circ)$ is the hadamard product operation, $a_{uv}^k = L^{'}(f_k(\mathbf{e}_u^*, \mathbf{e}_v^*) - o_{uv}^k)$ is a scalar. $\mathbf{r}^k = \frac{\partial f_k(\mathbf{e}_u^*, \mathbf{e}_v^*)}{\partial(\mathbf{e}_u^* \circ \mathbf{e}_v^*)}$. As $\mathbf{r}^k$ denotes the derivative of a scalar to a vector, it is also a vector. $\forall\, k \in \{1, 2, \ldots, K\}$, $a_{uv}^k \mathbf{r}^k$ determines the updating magnitude and direction of the vector $\mathbf{e}_u^* \circ \mathbf{e}_v^*$. We can see that the gradients from all behaviors are coupled, and they jointly optimize the same vector $\mathbf{e}_u^* \circ \mathbf{e}_v^*$, which leading to gradient conflicts. As a result, the harmful information coupled in the input affects the learning of the target behavior information in the training process, leading to negative transfer.

A.1.2 The Gradient Issue in Transfer-based MTL. As we have claimed the shortcomings of the classical coupled-input MTL in Appendix A.1.1, we apply a decoupled input for the existing transfer structure of multi-behavior learning framework. For any behavior k, we have:

$$\mathbf{h}_u^k = \mathbf{e}_u^k + \sum_{k_t=1}^{k-1} g_k(\mathbf{e}_u^{k_t}), \mathbf{h}_v^k = \mathbf{e}_v^k + \sum_{k_t=1}^{k-1} g_k(\mathbf{e}_v^{k_t}),$$

where k_t is the upstream behavior relative to k, i.e., $k_t < k$. For analytical convenience, we define $g_k(\cdot)$ as a linear function. Further, we take $(\mathbf{e}_u^t, \mathbf{e}_v^t)$ as input $(1 \le t \le K)$, and the loss function is as follows:

$$\mathcal{L}_{uv} = \sum_{k=1}^{K} L(f_k(\mathbf{h}_u^k, \mathbf{h}_v^k) - o_{uv}^k),$$

where $L(\cdot)$ is the loss function, and $f_k(\cdot)$ is the predictive function in MTL models. Then we have:

$$\begin{aligned}\frac{\partial \mathcal{L}_{uv}}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)} &= \sum_{k=1}^{K} \frac{\partial f_k(\mathbf{h}_u^k, \mathbf{h}_v^k)}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)} * L^{'}(f_k(\mathbf{h}_u^k, \mathbf{h}_v^k) - o_{uv}^k) \\ &= \sum_{k=1}^{K} a_{uv}^k \sum_{\substack{1 \le k_i \le k \\ 1 \le k_j \le k}} \frac{\partial f_k(\mathbf{e}_u^{k_i}, \mathbf{e}_v^{k_j})}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)},\end{aligned}$$

where $a_{uv}^k = L^{'}(f_k(\mathbf{h}_u^k, \mathbf{h}_v^k) - o_{uv}^k)$ is a scalar. It can be seen that in the process of optimization, for any behavior (e.g., the t-th behavior), the representations of other behaviors are directly coupled into the calculation, leading to gradient conflicts when updating $\mathbf{e}_u^t \circ \mathbf{e}_v^t$.

Last but not least, we further analyze $\mathbf{h}_u^k$:

$$\mathbf{h}_u^k = \mathbf{e}_u^k + \sum_{k_t=1}^{k-1} g_k(\mathbf{e}_u^{k_t}) = \mathbf{e}_u^k + \sum_{k_t=1}^{k-1} g_k(a_{k_t k} * \mathbf{e}_u^k) + \sum_{k_t=1}^{k-1} g_k(b_{k_t k} * \mathbf{e}_u^{\perp}),$$

where $a_{k_t k} = \frac{\mathbf{e}_u^{k_t} \cdot \mathbf{e}_u^k}{|\mathbf{e}_u^k||\mathbf{e}_u^k|}$ and $b_{k_t k} = \sqrt{|\mathbf{e}_u^{k_t}|^2 - (a_{k_t k}|\mathbf{e}_u^k|)^2}$ are constants. $\mathbf{e}_u^{\perp} \cdot \mathbf{e}_u^k = 0$. We can find that the behavioral representations transferred from upstream to downstream contains components orthogonal to the downstream representation. Therefore, harmful information is transferred from upstream to downstream, which affects the learning of target behavior information and leads to negative transfer. The derivation for the case of item v is similar.

A.1.3 Superiority of Our Proposed PTN. In this part, we analyze how the PTN module solves the above gradient conflicts problem, thus further solving negative transfer. According to Equation (12), we apply a decoupled input, and have only enhanced the user side:

$$\begin{aligned}\mathbf{g}_u^k &= \mathbf{e}_u^k + \alpha * \sum_{k_i=1}^{k-1} \mathbf{g}_{u,sha}^{k_i k} + \beta * \sum_{k_j=k+1}^{K} \mathbf{g}_{u,re}^{k k_j} \\ &= \mathbf{e}_u^k + \alpha * \sum_{k_i=1}^{k-1} \psi_k(c_{k_i k}^1 * \mathbf{e}_u^k) + \beta * \sum_{k_j=k+1}^{K} \psi_k(c_{k k_j}^2 * \mathbf{e}_u^k),\end{aligned}$$

where $c_{k_i k}^1 = \frac{\mathbf{e}_u^{k_i} \cdot \mathbf{e}_u^k}{|\mathbf{e}_u^k||\mathbf{e}_u^k|}$ and $c_{k k_j}^2 = \frac{(\mathbf{e}_u^k - \frac{\mathbf{e}_u^k \cdot \mathbf{e}_u^{k_j}}{|\mathbf{e}_u^{k_j}||\mathbf{e}_u^{k_j}|} \mathbf{e}_u^{k_j}) \cdot \mathbf{e}_u^k}{|\mathbf{e}_u^k||\mathbf{e}_u^k|}$ are constants. $k \in \{1, 2, \cdots, K\}$ and we set that $\sum_a^b = 0$ when $a > b$. $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are coefficients representing the weights of shared information and unique information. $\psi_k(\cdot)$ represents the message propagation and aggregation process of the Extraction Network under behavior k. We can see that $\mathbf{g}_u^k$ can be viewed as a mapping of $\mathbf{e}_u^k$. Then, we take $(\mathbf{e}_u^t, \mathbf{e}_v^t)$ as input, and have the following derivation:

$$\begin{aligned}\frac{\partial \mathcal{L}_{uv}}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)} &= \sum_{k=1}^{K} \frac{\partial L(f_k(\mathbf{g}_u^k, \mathbf{e}_v^k) - o_{uv}^k)}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)} \\ &= \sum_{k=1}^{K} \frac{\partial f_k(\mathbf{g}_u^k, \mathbf{e}_v^k)}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)} * L^{'}(f_k(\mathbf{g}_u^k, \mathbf{e}_v^k) - o_{uv}^k) \\ &= \sum_{k=1}^{K} a_{uv}^k \frac{\partial f_k(\mathbf{e}_u^k, \mathbf{e}_v^k)}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)} = a_{uv}^t \frac{\partial f_t(\mathbf{e}_u^t, \mathbf{e}_v^t)}{\partial(\mathbf{e}_u^t \circ \mathbf{e}_v^t)},\end{aligned}$$

where $a_{uv}^k = L^{'}(f_k(\mathbf{g}_u^k, \mathbf{e}_v^k) - o_{uv}^k)$ is a scalar. We can find that $\forall t \in \{1, 2, \cdots, K\}$, the gradient of each behavior optimizes along the direction of their respective input (e.g., the gradient of the t-th behavior optimizes $\mathbf{e}_u^t \circ \mathbf{e}_v^t$ independently), so that the gradient conflicts problem is successfully solved. Meanwhile, since the orthogonal components (Illustrated in Appendix A.1.2) with each behavioral input are removed by the projection mechanism, the negative transfer is well solved.

A.2 Explanation of the Re-projection

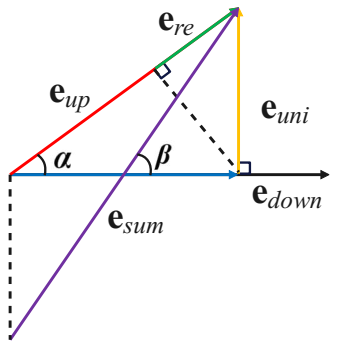


Figure 5: The illustration of the Re-projection.

In this part, we illustrate why we re-projects the unique representation back to the upstream representation. Compared with $\mathbf{e}^{up}$, $\mathbf{e}^{uni}$ lacks the shared information of upstream and downstream, and directly adding them up will destroy the correlations between upstream and downstream behavioral representations. A more figurative expression is shown in Figure 5, according to the parallelogram rule, we have $\mathbf{e}^{sum} = \mathbf{e}^{up} + \mathbf{e}^{uni}$. Furthermore, it can be clearly seen in the figure that $\beta > \alpha$ (i.e. $\mathbf{e}^{sum}$ tends to be more orthogonal to $\mathbf{e}^{down}$). Thus, we propose the re-projection method that refines $\mathbf{e}^{re}$ from $\mathbf{e}^{uni}$, making the final representation pay more attention to unique information without affecting the correlations between behaviors.

A.3 Explanation of Non-sampling Learning Loss

In this part, we illiterate non-sampling learning loss in detail. Before the derivation, we claim that:

$$\hat{o}_{uv}^k = (\mathbf{e}_u^k)^T \cdot diag\left(\mathbf{e}_r^k\right) \cdot \mathbf{e}_v^k,$$

which has been proposed in Section 4.1.3.

First, the original loss of GHCF is:

$$\begin{aligned}\mathcal{L}_k &= \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}^k \left(o_{uv}^k - \hat{o}_{uv}^k\right)^2 \\ &= \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}^k \left((o_{uv}^k)^2 - 2o_{uv}^k\hat{o}_{uv}^k + (\hat{o}_{uv}^k)^2\right).\end{aligned}$$

The time complexity of computing of the loss is $O(|\mathbf{U}||\mathbf{V}|d)$. The loss is simplified by the following steps: Because $\mathbf{V} = \mathbf{V}^{k+} + \mathbf{V}^{k-}$, $\forall v \in \mathbf{V}^{k+}, o_{uv}^k = 1$ and $\forall v \in \mathbf{V}^{k-}, o_{uv}^k = 0$, we have:

$$-2\sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}^k o_{uv}^k \hat{o}_{uv}^k = -2\sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^{k+}} c_{uv}^{k+}\hat{o}_{uv}^k + 0.$$

Thus, after eliminating the constant value, we have:

$$\tilde{\mathcal{L}}_k = -2\sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^{k+}} c_{uv}^{k+}\hat{o}_{uv}^k + \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}^k(\hat{o}_{uv}^k)^2.$$

Besides, we have:

$$\begin{cases}\sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}^k(\hat{o}_{uv}^k)^2 = \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^{k+}} c_{uv}^{k+}(\hat{o}_{uv}^k)^2 + \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^{k-}} c_{uv}^{k-}(\hat{o}_{uv}^k)^2, \\ \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_{uv}^{k-}(\hat{o}_{uv}^k)^2 = \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^{k+}} c_{uv}^{k-}(\hat{o}_{uv}^k)^2 + \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}^{k-}} c_{uv}^{k-}(\hat{o}_{uv}^k)^2.\end{cases}$$

Rearranging $\tilde{\mathcal{L}}_k$ and simplify c_{uv} to c_v, we can get:

$$\begin{aligned}\tilde{\mathcal{L}}_k &= \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}_u^{k+}}\left(\left(c_v^{k+} - c_v^{k-}\right)(\hat{o}_{uv}^k)^2 - 2c_v^{k+}\hat{o}_{uv}^k\right) + \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}} c_v^{k-}(\hat{o}_{uv}^k)^2 \\ &= \sum_{u\in\mathbf{U}}\sum_{v\in\mathbf{V}_u^{k+}}\left(\left(c_v^{k+} - c_v^{k-}\right)(\hat{o}_{uv}^k)^2 - 2c_v^{k+}\hat{o}_{uv}^k\right) \\ &\quad + \sum_{i=1}^{d}\sum_{j=1}^{d}\left(\left(\mathbf{e}_{r,i}^k\mathbf{e}_{r,j}^k\right)\left(\sum_{u\in\mathbf{U}}\mathbf{e}_{u,i}^k\mathbf{e}_{u,j}^k\right)\left(\sum_{v\in\mathbf{V}} c_v^{k-}\mathbf{e}_{v,i}^k\mathbf{e}_{v,j}^k\right)\right),\end{aligned}$$

where $\mathbf{V}_u^{k+}$ denotes the interacted items of user u under the behavior k. The complexity of the final loss is $O\left((|\mathbf{U}| + |\mathbf{V}|)d^2 + |\mathbf{V}^{k+}|d\right)$. Since $|\mathbf{V}^{k+}| \ll |\mathbf{U}||\mathbf{V}|$, the complexity of the final loss is much more less than the original one.

We apply the Non-sampling Learning Loss in our model. And the time complexity of the GNN part is $O\left((L + L_{tr})|\mathcal{E}|d\right)$. Since $(|\mathbf{U}| + |\mathbf{V}|) << |\mathcal{E}|$ and $\sum_{k=1}^{K}|\mathbf{V}^{k+}|d = |\mathcal{E}|d$, the time complexity of our model mainly comes from the GNN part.

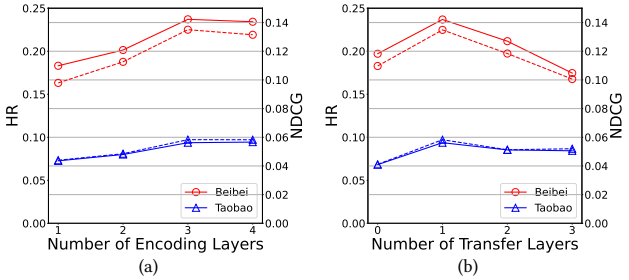


Figure 6: Impact of Layers. The solid line and the dotted line represent HR and NDCG, respectively.

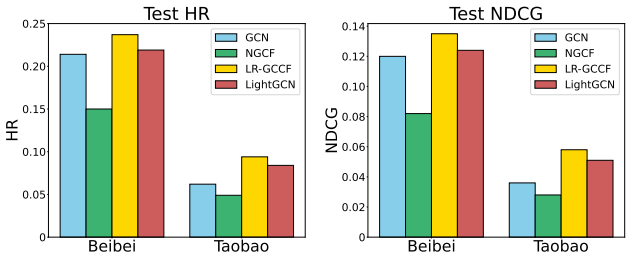


Figure 7: Impact of GCN aggregators.

A.4 Hyper-parameter Analysis

A.4.1 Impact of layers. To investigate whether our proposed HPMR can benefit from information propagation, we vary the number of encoding and transfer layers.

Number of Encoding Layers. As shown in Figure 6 (a), we can see that by adjusting the depth of Encoding Network from one to three, the performances on both Beibei and Taobao datasets are improved. Generally, three propagation layers are sufficient to capture the interactive information under each behavior. If we increase the number of layers further, the model's performance flattens out or even starts to decline. A probable reason is that deeper propagation may introduce noise and lead to overfitting.

Number of Transfer Layers. In order to ascertain the impact of the layer of Extraction Network, we search the transfer layer numbers in the range of [0, 1, 2, 3]. Besides, as shown in Figure 6 (b), we can find that the model's performance reaches its peak when the number of layers is one. And with the increase of layers, the performance gradually decreases. This may be because the shared representation that is transferred has obtained enough propagating information in Encoding Network. When it is transferred downstream, too deeper propagation will lead to overfitting.

A.4.2 Impact of GNN aggregators. We investigate the impact of different GNN aggregators i.e. GCN [22], NGCF [35], LR-GCCF [9] and LightGCN [16]. The models with different aggregators are compared in Figure 7. We can find that LR-GCCF performs the best on Beibei and Taobao among the four aggregators. One possible reason might be that Beibei and Taobao contain multiple types of closely correlated behaviors, which have high requirements on the fitting ability of the model. So the introduction of a nonlinear activation function better facilitates the model to fit these datasets.