Enhanced Pre-training for Recommendation via Hypergraph Structural Entropy

JINGYUN ZHANG, Beihang University, China
HAO PENG*, Beihang University, China
MINGDAI YANG, University of Illinois at Chicago, USA
PHILIP S. YU, University of Illinois at Chicago, USA

Research on recommender systems plays a crucial role in alleviating information overload amid the current proliferation of data while diminishing user decision-making and transaction costs within intricate environments. The prevailing recommendation models currently rely on graph-based methods, such as GCN, GAT, HGNN, etc., which are constrained by the sparsity of training data and the underutilization of graph structures. In this work, we present EPRHSE, an **E**nhansed **P**re-training framework for **R**ecommendation based on **H**ypergraph **S**tructural **E**ntropy, which encodes the topology of the recommender system. We begin by designing two forms of pre-training tasks to capture the heterogeneous relationships among users or items. These pre-training tasks build multiple auxiliary task hypergraphs, compensate for the sparse interactions between users and items, and unveil latent information. Secondly, we introduce a new method for optimizing the hypergraph structure entropy. The method involves converting the hyperedge information in the hypergraph to form a high-dimensional encoding tree. Hypergraph structure entropy helps decode the essential structure of the recommendation bipartite graph and enables hierarchical clustering of users or items. Thirdly, we propose a hypergraph pooling training methodology incorporating pooling and unpooling layers into the hypergraph convolutional network to amalgamate high-order information. By transferring advanced community insights to primary users or items, the process of social diffusion is enhanced, consequently refining node embedding quality. Compared with thirteen representative recommendation approaches on five real datasets, comprehensive experiments demonstrate the advantages of the effectiveness of EPRHSE.

CCS Concepts: • **Information systems → Recommender systems**; • **Computing methodologies → Artificial intelligence**.

Additional Key Words and Phrases: Recommender System, Structure Entropy, Hypergraph Learning, Multitask Pre-training, Hypergraph Pooling

ACM Reference Format:
Jingyun Zhang, Hao Peng, Mingdai Yang, and Philip S. Yu. 2025. Enhanced Pre-training for Recommendation via Hypergraph Structural Entropy. 1, 1 (October 2025), 47 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*This is the corresponding author.

Authors' addresses: J. Zhang, School of Cyber Science and Technology, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing, 100191, China; email: zhangjingyun@buaa.edu.cn; H. Peng, School of Cyber Science and Technology, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing, 100191, China; email: penghao@buaa.edu.cn; Mingdai Yang, Department of Computer Science, University of Illinois at Chicago, IL 60607, USA; e-mail: myang72@uic.edu; P. S. Yu, Department of Computer Science, University of Illinois at Chicago, Chicago 60607, IL; email: psyu@uic.edu.

ACM XXXX-XXXX/2025/10-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Information and communication technologies have intensified the impact of information overload. The proliferation of multiple platforms disseminating the same content overwhelms individuals with abundant available information, making it challenging to utilize relevant information effectively [23, 38]. Due to a lack of control over information, individuals are prone to being overwhelmed by the sheer volume of information, leading to information fatigue, conflicts, stress, and anxiety, ultimately impacting productivity and innovation [37, 64, 74]. With the rapid advancement of e-commerce and social media, recommender systems are widely utilized in various industries such as e-commerce websites [13, 49], audio-visual platforms [17], and social media enterprises [30, 82]. Recommender systems leverage historical data on user interests, behaviors (purchases, searches, clicks, views, etc.), and item evaluations to recommend suitable products to users, filtering out essential information fragments. The core of recommender systems lies in retrieving items from user interaction history and context and modeling user interest preferences [47, 100]. Hence, researching recommender systems is of paramount importance in alleviating issues related to information overload [1, 2, 6], information cocoons [16, 46], reducing transaction costs for users when selecting products in complex big data environments, and enhancing the quality of user decision-making [2, 15].

Effective and accurate recommendations require thoroughly understanding, modeling, and analyzing user-item interaction data and user interests [24, 49, 89, 94]. Early methods rely on similarities between items or users, such as content-based filtering [45, 52, 66, 68] and collaborative filtering [18, 26, 72, 122]. The fundamental concept behind these methods is that individuals with similar preferences in the past are likely to have similar choices in the future. Whether based on user similarity or item similarity, these methods are constrained by simple low-dimensional features, e.g., rating information, attribute information, etc., as well as primary classification or ranking algorithms [5, 20, 22], limiting their expressive power. While methods based on matrix factorization [39, 61, 71] and logistic regression [65, 81, 92] have improved handling and generalization in sparse data scenarios by learning latent feature vectors from social networks, these representative models are ineffective in making recommendations when lacking user historical behavioral data. Graph neural network models [14, 24, 101] capture intricate user-item topological structures to learn item attributes and optimize simple collaborative filtering mechanisms. Although these models are considered sufficient, they require large amounts of user-item historical interaction data, posing a risk to their generalization in the cold-start scenario [44, 94]. Therefore, pre-training is essential for recommender systems. Training on diverse datasets enables the model to learn more general features, compensating for the limited data available for specific recommendation tasks.

Although Graph Neural Networks (GNNs) can capture node information and partial topological structure, they rely on pairwise interactions and overlook complex higher-order relationships. Recent studies have incorporated hypergraphs into recommender systems and developed hypergraph convolutional network-based models [43, 88, 106]. Additionally, since incorporating social information plays a crucial role in cold-start recommendation scenarios, attention mechanisms are commonly employed to aggregate information from different graphs and domains [8, 87, 120]. Consequently, the "*pre-training + hypergraph + attention*" paradigm has become a prominent research focus, applied to various specific tasks such as sequential recommendation, point-of-interest (POI) recommendation, and group recommendation. For instance, UPRTH [108] proposes a Unified Pre-training framework for Recommendation via Task Hypergraphs. To establish a unified learning framework capable of handling the diverse requirements and nuances of various pretext tasks, UPRTH incorporates task hypergraphs to generalize pretext tasks as hyperedge prediction. A Transitional Attention layer (TA layer) is designed to learn the relevance between each pretext task and recommendation discrimination. However, we argue that such models still face the following challenges:

- **In the context of hypergraph learning**, current research has not fully leveraged the **topological structure of hypergraphs**, which is a critical factor for effective recommendations [89]. Existing studies further

emphasize that the potential of hypergraphs in modeling higher-order user relationships remains underutilized [114]. Prior work has also demonstrated that topological structure significantly influences information retrieval, processing, and decision-making [59]. Consequently, effectively learning the topological structure of hypergraphs remains a key research challenge.

- **In the context of model architecture**, simply employing hypergraph neural networks fails to capture **global dependency relationships** adequately. First, hypergraph convolutional networks (HGCNs) rely excessively on local high-order neighborhood structures, limiting their ability to model social influence diffusion processes across the global network [98]. Second, both indirect social connections and historical user-item interactions play significant roles in identifying user preference groups [55]. Therefore, capturing the dependency relationships between direct neighbors and learning the group representations from the global network [29] for target users or items is crucial.
- The **complexity and abstraction introduced by information integration** remain a persistent challenge. Most existing models following the "pre-training + hypergraph + attention" paradigm rely on attention mechanisms or similar strategies to integrate heterogeneous types of social information. However, this integration process is inherently built upon diverse GCN architectures, which introduces excessive complexity and abstraction, thereby undermining model interpretability. Taking UPRTH as an example, it employs an attention mechanism to fuse auxiliary-task embeddings into the hyperedge embeddings of the primary-task hypergraph. Yet, this integration is intrinsically subjective, since the hyperedge embeddings in the item graph are not semantically equivalent to the user embeddings derived from auxiliary tasks. Specifically, while hyperedge embeddings encode shared co-purchasing patterns among items, they fail to comprehensively capture user-specific attributes.

In this work, we propose EPRHSE, a novel **E**nhanced **P**re-training framework for **R**ecommendation based on **H**ypergraph **S**tructural **E**ntropy. Our framework systematically addresses three key challenges through innovative hypergraph learning techniques and architectural advancements. The proposed approach strategically incorporates auxiliary social information structures to enhance representation learning while avoiding non-interpretable cross-hypergraph information fusion. The framework operates through three main components. First, we design dual pre-training tasks that model users (items) as nodes and social commonalities as hyperedges, constructing multiple auxiliary task hypergraphs. This design effectively captures heterogeneous relationships among users (items), uncovers latent user/item information through auxiliary task constraints, and alleviates interaction sparsity issues. Second, building upon structural information theory [48], we develop a novel hypergraph structural entropy optimization method that generalizes homogeneous structural entropy to hypergraphs. This involves constructing and optimizing high-dimensional encoding trees to establish hierarchical node clusters, thereby improving information structure comprehension. Hierarchical partitioning helps capture both global and local dependencies, allowing our model to generate more informative user and item embeddings by reflecting group-level patterns while preserving individual-specific information. Third, we augment the conventional HGCN encoder with hypergraph pooling and unpooling layers. While the original encoder captures direct dependencies from primary and auxiliary task hypergraphs, these new layers extract global community representations from the encoding tree. Specifically, the pooling layer aggregates low-level node representations into high-level community representations, which are then propagated back through unpooling to refine individual node embeddings. This dual mechanism significantly enhances the modeling of social diffusion processes. This enhanced structured pre-training framework sufficiently leverages auxiliary social information to constrain representation learning while simultaneously reducing both model complexity and interpretability issues arising from multi-level hypergraph fusion.

We conduct extensive experiments on five datasets, Steam, XMrec-CN, XMrec-MX, XMrec-AU, and XMrec-BR, to demonstrate the effectiveness of EPRHSE. First, the overall experimental results indicate that EPRHSE

demonstrates superior overall performance compared to ten baseline models. Second, a series of ablation studies are conducted to analyze the rationale and effectiveness of the hypergraph pooling layer and the exclusion of hypergraph representation fusion (i.e., the TA layer) in EPRHSE. Thirdly, hyperparameter experiments analyze the sensitivity and selection of four parameters in EPRHSE on different datasets. Then, the experiment in the cold-start scenario further illustrates the high performance and stability of EPRHSE. Finally, we conduct computational efficiency analysis by comparing the time consumption and training curves of EPRHSE with baseline methods. The codes for all baseline models and EPRHSE, along with all datasets, are publicly accessible on GitHub[1].

The main contributions of this work are summarized as follows:

- A new Enhanced Pre-training framework for Recommendation based on Hypergraph Structural Entropy is proposed with high effectiveness and encodes the topology of recommender systems.
- A new hypergraph structure entropy optimization method is proposed to achieve hierarchical community division of users and items by transferring edge information from the node-hyperedge bipartite graph to the node adjacency matrix.
- A new global information aggregation method is developed, which improves the HGCN model architecture through hypergraph pooling and unpooling layers to learn better node representations.
- A novel design omitting the hypergraph fusion paradigm (like the TA layer used in UPRTH) is demonstrated to preserve recommendation accuracy while reducing model complexity and improving interpretability.
- A series of comparative analysis experiments show that EPRHSE achieves higher recommendation quality and strong stability even in the cold-start scenario.

The structure of this paper is as follows. Section 2 outlines the background and preliminaries of our work. In Section 3, we describe the technical details of the proposed framework, named EPRHSE. Section 4 presents the experimental setup, and Section 5 discusses the experiment's results. Section 6 provides an overview of related works. Finally, we conclude the paper in Section 7.

2 PRELIMINARY

In this section, we first provide the fundamental definition and concepts of the recommendation task (Subsection 2.1). Next, we elaborate on the hypergraph encoder's structure and information aggregation methods (Subsection 2.2). Then, we introduce the application of structural information theory in the recommendation framework and elaborate on the basic concepts of structural entropy (Subsection 2.3). Finally, we deliver a detailed description of the hypergraph fusion mechanism (TA layer), which will be added to EPRHSE in ablation studies for discussion (Subsection 2.4). The comprehensive list of the primary symbols used throughout this paper is presented in Table 1.

2.1 Recommendation Task

The recommendation task aims to determine the optimal ranking of items for each user based on the given user-item interaction information. This information includes two separate sets of nodes (a user set U and an item set I) and user-item interactive edges $E_{u,i}$. Therefore, the graph representing user-item interactions can be $G = (U, E_{u,i}, I)$. In a personalized recommendation task T_{rec} for a user, the goal is to predict a list of items $\{i_1, i_2, ..., i_m\}$ in graph G that the user has not yet interacted with. The higher the ranking, the more likely the user will interact with those items.

2.2 Hypergraph Encoder

The hypergraph is composed of a node set $\mathcal{V}$ and a hyperedge set $\mathcal{E}_{hyper}$, represented as $\mathcal{H} = (\mathcal{V}, \mathcal{E}_{hyper})$. Unlike a regular graph, its hyperedges can connect multiple nodes, indicating the relevance among a group of nodes

[1]https://github.com/SELGroup/EPRHSE

Table 1. Forms and interpretations of notations.

Symbol	Definition
$G; \mathcal{G}; \mathcal{H}$	User-item interaction graph; Homogeneous graph; Hypergraph.
$E_{u,i}; U; I; \mathcal{V}$	User-item interaction edge set; User set; Item set; Node set.
$C_1; C_2$	Preliminary community set; Final community set.
$\mathcal{E}_{hyper}; \mathcal{E}$	Hyperedge set; Homogeneous edge set.
$\mathcal{D}_{rec}$	Set of user-item interaction pairs in the training data.
$A_G; A_{\mathcal{H}}$	Bipartite graph matrix; Hypergraph adjacent matrix.
$i; u; v; e; c$	Item; User; Node; Hyperedge; Community.
$D_e; B_v$	Degree matrix of hyperedges; Degree matrix of nodes.
$\mathbf{E_u}; \mathbf{E_i}$	Representations of users; Representations of items.
$\mathbf{E_v}; \mathbf{E_e}$	Representations of nodes, Representations of hyperedges.
$\mathbf{E_c}$	Representations of communities.
$S_1; S_2$	Community division matrix for layer 1; Community division matrix for layer 2.
$d; d_e^m$	Embedding size; The degree of the m-th hyperedge.
$T_{rec}; T_a$	Recommendation task; Auxiliary task.
w_t	Attention diagonal matrix for the t-th auxiliary task T_{a_t}.
γ_1	The fusion hyperparameter between the main and auxiliary tasks.
γ_2	The fusion hyperparameter between the original and auxiliary task fusion embedding.
β_{hpu}	The weight hyperparameter of the previous embedding in hypergraph unpooling.
λ_{rec}	The hyperparameter to balance the losses between recommendation and auxiliary tasks.
λ_Θ	The hyperparameter for regularization.
$\mathcal{T}; \lambda$	Encoding tree; The root node of the encoding tree.
$\mathcal{T}_{mg}$	Encoding tree after Merging operator.
$\alpha; T_\alpha$	Node on encoding tree; Label of encoding tree node α.
$\alpha_i; \alpha^-$	i-th child node of encoding tree node α; Parent of encoding tree node α.
$d_i; g_\alpha$	Degree of node v_i; Number of cutting edges of encoding tree node α.
$vol(\mathcal{G}); vol(\alpha)$	Volume of Graph $\mathcal{G}$; Volume of encoding tree node α.
$H^{\mathcal{T}}(\mathcal{G})$	The structural entropy of $\mathcal{G}$ under encoding tree $\mathcal{T}$.
$H^{\mathcal{T}}(\mathcal{H}; \alpha)$	The structural entropy of subtree α in hypergraph $\mathcal{H}$ under encoding tree $\mathcal{T}$.
$H_k(\mathcal{G})$	The k-dimensional structure entropy.
$\Delta_{\mathcal{H}}^{Mg}$	Difference of Structural entropy after merging.
$\mathcal{T}_{mg}(\alpha, \beta)$	Merging operator between node α and node β.
Q	Maximum number of pairs in merge operator.

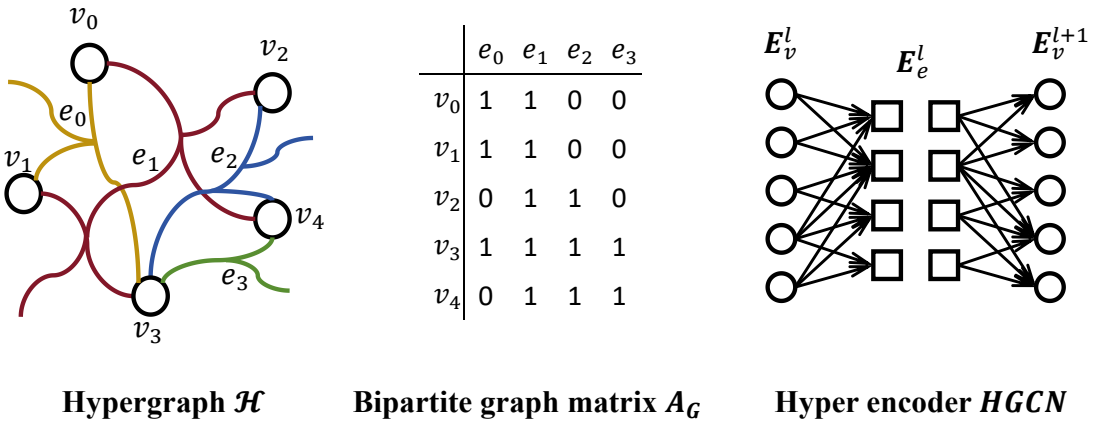


Fig. 1. An illustration of a hypergraph, bipartite matrix, and HGCN.

rather than just a pairwise relationship. Typically, the relationship between nodes and hyperedges in a hypergraph is characterized using a bipartite graph matrix A_G, specifically denoted as $A_G = (a_{i,j}) \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}_{hyper}|}$, where $a_{i,j}$ indicates whether node v_i and hyperedge e_j are connected. Hypergraph neural networks have been proposed for representation learning in hypergraph data. For each node in the hypergraph, the Hypergraph Convolutional Neural Network (HGCN) encoder iteratively updates its representation by combining information from its adjacent hyperedges. In contrast, the information from hyperedges is represented by combining information from adjacent nodes. This way, information is transferred through hyperedges to learn representations incorporating information from one-hop neighboring nodes, which can be propagated by stacking multiple encoders. The hypergraph encoder used in our model can be represented as:

$$\mathbf{E_v^{l+1}} = (\mathbf{D_e})^{-1} \cdot \mathbf{A_G} \cdot \mathbf{E_e^l} = (\mathbf{D_e})^{-1} \cdot \mathbf{A_G} \cdot (\mathbf{B_v})^{-1} \cdot \mathbf{A_G}^\top \cdot \mathbf{E_v^l}, \tag{1}$$

where $\mathbf{E_v^l}$ and $\mathbf{E_v^{l+1}}$ are the input and output embedding representations of the node set in a hypergraph encoder. A_G is the bipartite graph matrix of nodes and hyperedges, D_e is the degree matrix of hyperedges, and $\mathbf{E_e^l}$ represents the embedding representations of the set of hyperedges in the current iteration. Through dot product operations, the embedding representation of the node set is updated as the weighted sum of the embedding representations of its adjacent hyperedges. The more nodes a hyperedge is connected to, the smaller the influence on each node. Furthermore, the embedding representation of the set of hyperedges $\mathbf{E_e^l}$ can be obtained through the operation $(B_v)^{-1} \cdot {A_G}^\top \cdot \mathbf{E_{v,}^l}$, where B_v is the degree matrix of nodes in the hypergraph. Similarly, through dot product operations, the embedding representation of the set of hyperedges is updated as the weighted sum of the embedding representations of its connected nodes. The more hyperedges a node is connected to, the smaller the influence on each hyperedge.

2.3 Structural Information Theory

Structural information theory [48] is originally proposed for measuring the structural information contained within a graph. Specifically, this theory aims to calculate the structural entropy of the homogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which reflects its uncertainty when undergoing hierarchical division. In our work, the hierarchical partitions are represented by a tree structure known as the encoding tree. We introduce encoding trees and k-dimensional structural entropy in undirected graphs below.

Encoding tree. An encoding tree of a graph $\mathcal{G}$ is defined as a rooted tree $\mathcal{T}$ with the following properties:

(1) For each node α in the encoding tree $\mathcal{T}$, there is a subset $T_\alpha \in \mathcal{V}$ of vertices in the graph $\mathcal{G}$ corresponding to it.
(2) For the root node λ in the encoding tree, $T_\lambda = \mathcal{V}$.
(3) The children of node α are denoted as α_i and sorted from left to right as i increases. The parent node of α_i is denoted as $\alpha_i^- = \alpha$.
(4) If node α has L children, then the vertex subset T_{α_i} of child nodes is mutually exclusive, and $T_\alpha = \cup T_{\alpha_i}$.
(5) Each leaf node v in the tree, v corresponds to a single vertex in the vertex set $\mathcal{V}$ in the graph $\mathcal{G}$.

The k-dimensional encoding tree has a height of k (the height of the root node is 0). Intuitively, the encoding tree embodies the hierarchical community division of graph vertices; the parent node is a large community, and the child nodes are small communities in the large community.

Structure entropy. The structural information of the homogeneous graph $\mathcal{G}$ determined by the encoding tree $\mathcal{T}$ is defined as:

$$H^{\mathcal{T}}(\mathcal{G}) = -\sum_{\alpha \in \mathcal{T}, \alpha \neq \lambda} \frac{g_\alpha}{vol(\mathcal{G})} log \frac{vol(\alpha)}{vol(\alpha^-)}, \tag{2}$$

where $vol(\mathcal{G})$ is the sum of the degrees of all vertices in the graph $\mathcal{G}$. $vol(\alpha)$ is the volume of T_α and is the sum of the degrees of all vertices in the vertex subset T_α. g_α is the sum of weights of all edges from vertex subset T_α to vertex subset $\mathcal{V}/T_\alpha$, which can be understood as the total weight of the edges from the vertices outside the vertex subset T_α to the vertices inside the vertex T_α, or the total weight of the cut edges. $\frac{g_\alpha}{vol(\mathcal{G})}$ represents the probability that the random walk enters T_α. The structural entropy $H(\mathcal{G})$ of graph $\mathcal{G}$ is the minimum $H^{\mathcal{T}}(\mathcal{G})$. Let $\mathcal{T}_k$ be encoding trees whose height is not greater than k, then the k-dimensional structural entropy of $\mathcal{G}$ is defined as follows:

$$H_k(\mathcal{G}) = min(H^{\mathcal{T}_k}(\mathcal{G})). \tag{3}$$

Furthermore, one-dimensional structural entropy is special as there are only root nodes and leaf nodes in the encoding tree of one layer. All the vertices in the graph $\mathcal{G}$ belong to a large community λ under the one-dimensional encoding tree, which is unique in terms of community division so that the one-dimensional structural entropy can be directly expressed as:

$$H_1(\mathcal{G}) = -\sum_{i=1}^{n} \frac{d_i}{vol(\mathcal{G})} log \frac{d_i}{vol(\mathcal{G})}, \tag{4}$$

where d_i is the sum of weights of all edges connected to vertex v_i in graph $\mathcal{G}$ and is called the degree of vertex v_i. One-dimensional structural entropy measures the uncertainty of graph $\mathcal{G}$ without layering.

2.4 Hypergraph Fusion Mechanism (TA layer)

In this section, we introduce a hypergraph fusion mechanism, analogous to the TA layer in UPRTH [108], not as a component of our proposed model but as a variant employed for comparative analysis in subsequent discussions and experiments. This approach posits that, during the fusion process, each auxiliary task exerts varying levels of influence on the recommendation gains; hence, it is designed to adaptively learn the degree of correlation between auxiliary tasks and the main task. This transitional layer allocates appropriate attention to each auxiliary task, enabling the fusion of embeddings learned from auxiliary tasks into the embeddings of the main task via a single round of HGCN, thus facilitating effective knowledge transfer. To illustrate, taking the main task hypergraph "items bought by the same user" as an example, we represent the item node embeddings learned from the primary task as $\mathbf{E_i}$. As depicted in Figure 2, through the aggregation of node representations in HGCN, the initial hyperedge embeddings can be denoted as $\mathbf{E_e}$. Since in this main task hypergraph, hyperedges represent bought by the same user, $\mathbf{E_e}$ is equivalent to the commonality of the initial embeddings representing user purchases. Subsequently,

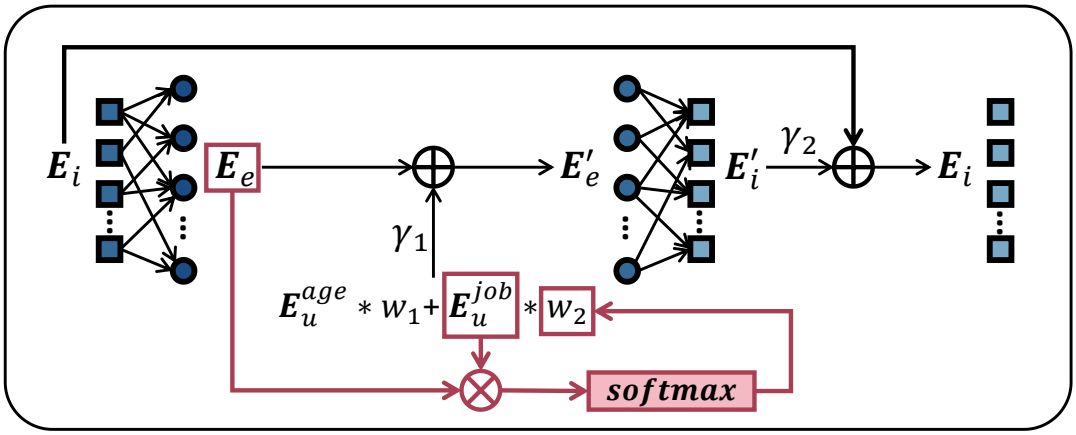


Fig. 2. An illustration of TA layer.

we compute the attention between the user features $\mathbf{E}_\mathbf{u}^{\mathbf{job}}$ ($\mathbf{E}_\mathbf{u}^{\mathbf{age}}$) learned from the auxiliary task and the initial hyperedge embeddings $\mathbf{E_e}$, generating auxiliary task hyperedge embeddings. Take E_u^{job} as an example:

$$w_2 = Softmax(\frac{diag(\mathbf{E}_\mathbf{u}^{\mathbf{job}} \cdot (\mathbf{E_e})^\top)}{\sqrt{d}}), \mathbf{E}_\mathbf{e}^\mathbf{a} = \mathbf{w_1}\mathbf{E}_\mathbf{u}^{\mathbf{age}} + \mathbf{w_2}\mathbf{E}_\mathbf{u}^{\mathbf{job}}, \tag{5}$$

where $\mathbf{E}_\mathbf{e}^\mathbf{a}$ represents the embedding of auxiliary task hyperedges, which is obtained by weighting the attention of user features from all the auxiliary tasks. $\mathbf{E}_\mathbf{u}^{\mathbf{job}}$ denotes the user features from the auxiliary task "users with same jobs", and $\mathbf{E}_\mathbf{u}^{\mathbf{age}}$ denotes the user features from the auxiliary task "users with same ages". d signifies the feature dimension. w_1 and w_2 are attention diagonal matrixes, representing the attention allocated to the user for each auxiliary task. Subsequently, the initial hyperedge embedding is fused with the auxiliary task hyperedge embedding through addition:

$$\mathbf{E_e}' = \mathbf{E_e} + \gamma_1 \mathbf{E}_\mathbf{e}^\mathbf{a} = \mathbf{E_e} + \gamma_1(\mathbf{w_1}\mathbf{E}_\mathbf{u}^{\mathbf{age}} + \mathbf{w_2}\mathbf{E}_\mathbf{u}^{\mathbf{job}}), \tag{6}$$

where $\mathbf{E_e}'$ denotes the fused hyperedge embedding, and γ_1 represents the fusion hyperparameter. Finally, we update the item embeddings to $\mathbf{E}_\mathbf{i}'$ by aggregating hyperedge embeddings in HGCN and fuse it into the original item embedding $\mathbf{E_i}$ with fusion hyperparameter γ_2:

$$\mathbf{E_i} = \mathbf{E_i} + \gamma_2 \mathbf{E}_\mathbf{i}'. \tag{7}$$

The TA layer includes two hyperparameters, γ_1 and γ_2, which control the weights in the two-step embedding fusion process, one for hyperedges and one for nodes.

3 THE PROPOSED MODEL

This section will introduce the main component of the recommender framework EPRHSE. This new work aims to learn better representations of fused structures to enhance recommender systems. As shown in Figure 3, EPRHSE consists of three key modules: pre-training process, hypergraph structural entropy module, and hypergraph pooling module. Specifically, the pre-training process (Section 3.1) defines two types of auxiliary tasks. It outlines the model's entire pre-training and recommendation process, utilizing an optimized HGCN layer to learn item or user node embeddings in different hypergraphs. Then, the hypergraph structural entropy module and the hypergraph pooling module collectively elaborate on how HGCN is optimized from the perspectives of

hypergraph structural learning and global information capture. This is the first work to extend the structural information theory to hypergraph, referred to as hypergraph structural entropy (Section 3.2). The node-hyperedge bipartite graph is converted into an adjacency matrix of nodes, and the hypergraph structural entropy is designed by transferring hyperedge information. Subsequently, the optimal high-dimensional community division is formed under the guidance of the principle of structural entropy minimization. The Hypergraph pooling module (Section 3.3) adds two pooling layers and two unpooling layers after the Hypergraph encoder HGCN, respectively. The group features are integrated into node embeddings for recommendation by aggregating and transferring community embeddings. Finally, all modules are integrated to optimize the recommendation task (Section 3.4), and the time complexity of the model was analyzed in detail (Section 3.5).

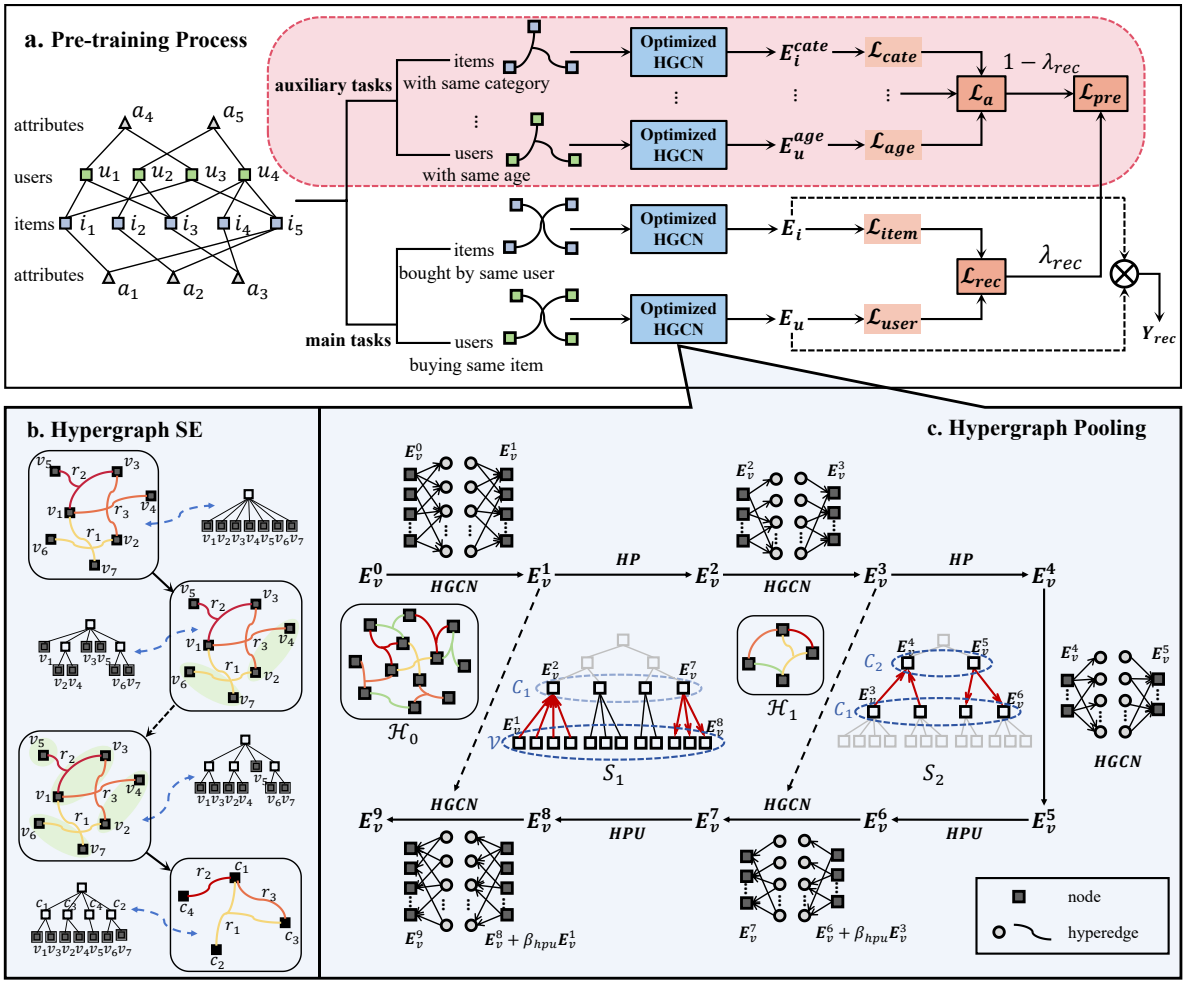


Fig. 3. The overall framework of EPRHSE.

3.1 Pre-training Process

This subsection will introduce the step-by-step methodology of the entire pre-training process. Pre-training aims to acquire prior knowledge from auxiliary and main tasks and build better user/item embeddings for downstream recommendation models. Throughout the pre-training process, we learn two embedding matrices, $\mathbf{E_i}$ and $\mathbf{E_u}$, where rows denote the representations of items and users.

Hypergraph construction. We begin by constructing a hypergraph for each auxiliary task to unify the handling of various auxiliary tasks. These tasks may involve specific attributes, such as age, occupation, price, category, etc., about users and items and may also include relationships, such as substitutes and complements. Using users' age attributes as an example, we represent users as nodes in the hypergraph, connecting groups of users within the same age bracket with hyperedges. This step yields the hypergraph illustrated in the upper part of Figure 3.a, denoted as "users same age". Similarly, we can also construct auxiliary task hypergraphs, such as "items with the same category, "items with the same rate, "items bought together," "items compared together," and "users with the same job. More details of auxiliary tasks for different datasets are presented in Table 2. Then, two other hypergraphs are constructed for the main task, with nodes representing users and items separately with direct item-user interactions, denoted as "users buying same item" and "items bought by same user", as shown in the lower part of Figure 3.a.

Hypergraph learning. All auxiliary task hypergraphs and the main task hypergraphs are fed into the optimized HGCN layer for training, producing the corresponding representations for item and user nodes. For instance, $\mathbf{E_i}$ denotes the embeddings of item nodes in the main task hypergraph "items bought by same user", $\mathbf{E_u}$ denotes the embeddings of user nodes in the main task hypergraph "users buying same item", $\mathbf{E_i^{cate}}$ represents the embeddings of item nodes in the auxiliary hypergraph for "items with same category," and $\mathbf{E_u^{age}}$ represents the embeddings of user nodes in the auxiliary hypergraph for "users with same age". The optimized HGCN incorporates the learning of hypergraph topology (as discussed in Subsection 3.2) and captures global relationships (as discussed in Subsection 3.3), enhancements to the initial HGCN, which will be elaborated upon in subsequent chapters.

Pre-training loss and recommendation. The representations above are utilized as optimization objectives during pre-training, with individual losses computed for each. The losses from auxiliary tasks are aggregated to form the auxiliary loss $\mathcal{L}_a$, while those from main tasks are combined to derive the recommendation loss $\mathcal{L}_{rec}$. The final pre-training loss is obtained by applying a weighted combination of these two components. The recommendation sequence is generated during the recommendation phase by comparing the two main task representations, $\mathbf{E_i}$ and $\mathbf{E_u}$. The detailed calculation of the loss function will be introduced in Subsection 3.4.

3.2 Hypergraph Structural Entropy

Structural information theory shows stronger adaptability in learning graph structure and clustering. However, this method is limited to simple homogeneous graphs and multi-relational graphs. To solve the challenges of constructing encoding trees and dividing communities on hypergraphs, we first transform the hyperedge information in the hypergraph's bipartite graph matrix to build the node adjacency matrix. Then, we construct the optimal encoding tree for the hypergraph to partition the node community effectively.

Hypergraph structural entropy. The definition of traditional structural entropy is given in Section 2.3. After constructing the main and auxiliary task hypergraphs, our objective is to classify users/items with close connections into the same community using structural information theory. Structural information theory determines structural uncertainty based on random walks of nodes in the graph through connected edges. Since a hyperedge in a hypergraph can connect multiple nodes, it expresses the relationship between groups rather than one-to-one relationships. This is the most significant difference between hypergraphs and homogeneous

graphs/multi-relationship graphs. Therefore, the key to generalizing the structural entropy concept to a hypergraph is to transform the group relationships (i.e., hyperedges) in the hypergraph into abstract one-to-one relationships between nodes (i.e., node adjacency matrix) while preserving the hypergraph's characteristics. We perform the following operations on the bipartite graph matrix A_G of the hypergraph to obtain the node adjacency matrix $A_{\mathcal{H}}$:

$$A_{\mathcal{H}} = A_G \cdot (D_e)^{-1} \cdot A_G{}^{\top} = (\sum_{m=1}^{|\mathcal{E}_{hyper}|} a_{i,m} \cdot a_{j,m} \cdot \frac{1}{log_2 d_e^m}), \tag{8}$$

where D_e represents the hyperedge degree matrix, the same as Eq. 1. d_e^m represents the degree of the m-th hyperedge. And $a_{i,m}$ represents whether there is a connection relationship between node v_i and hyperedge e_m, which is the same as in Section 2.2. The above formula assigns a value to the adjacency relationship of nodes by calculating the weighted sum of the hyperedges shared between two nodes. The function of $(D_e)^{-1}$ is to amplify the influence of hyperedges with fewer connected nodes.

After obtaining the adjacency matrix of the hypergraph node, the hypergraph structure entropy can be calculated through Eq. 2- 4. Specifically, the degrees of all nodes are represented by the corresponding row sums of the adjacency matrix:

$$(d_1, d_2, ..., d_N)^{\top} = (A_{\mathcal{H}} - diag(A_{\mathcal{H}})) \cdot \mathbf{1}_{|\mathcal{V}|}, \tag{9}$$

where $diag(\cdot)$ represents taking the diagonal elements of the matrix, $|\mathcal{V}|$ is the num of nodes in the hypergraph $\mathcal{H}$, and $\mathbf{1}_{|\mathcal{V}|}$ is a column vector of length $|\mathcal{V}|$ that is all 1, i.e. $(1, 1, ..., 1)^{\top}$. In the calculation process of the above formula, the adjacent items from the node itself to itself are removed.

Optimized encoding tree. We construct the optimal three-dimensional encoding tree by constructing the optimal two-dimensional encoding tree twice. After forming the optimal two-dimensional encoding tree, we use these community nodes as hypergraph nodes to construct a two-dimensional encoding tree again. The construction process of the two-dimensional encoding tree is as follows. The initial one-dimensional encoding tree represents the simplest two-level structure, where the leaf nodes in the graph are directly connected to the root node. The Merge operator combines the nodes, and a greedy search strategy is employed to construct an optimal encoding tree. The Merge operator combines two subtrees under the same parent node, resulting in a single subtree. This is visualized in the graph as the merging of two small communities. In the optimal encoding tree, the graph structure exhibits minimal uncertainty, and the nodes achieve a balanced and stable state, leading to the optimal partitioning of nodes. Recording $\mathcal{T}_{mg}(\alpha, \beta)$ as the encoding tree after $\mathcal{T}$ runs $Mg(\mathcal{T}; \alpha, \beta)$, the difference in structural entropy of the graph $\mathcal{G}_{hyper}$ determined by the two encoding trees $\mathcal{T}_{mg}(\alpha, \beta)$ and $\mathcal{T}$ is:

$$\Delta_{\mathcal{H}}^{Mg}(T; \alpha, \beta) = H^{\mathcal{T}_{mg}(\alpha,\beta)}(\mathcal{H}; \alpha) + \sum_{\delta^-=\alpha} H^{\mathcal{T}_{mg}(\alpha,\beta)}(\mathcal{H}; \delta)) - H^{\mathcal{T}}(\mathcal{H}; \alpha) - H^{\mathcal{T}}(\mathcal{H}; \beta) - \sum_{\delta^-=\alpha \text{ or } \delta^-=\beta} H^{\mathcal{T}}(\mathcal{H}; \delta), \tag{10}$$

where $H^{\mathcal{T}}(\mathcal{H}; \alpha)$ is the structural information of subtree α, $H^{\mathcal{T}}(\mathcal{H}; \beta)$ is the structural information of subtree β, $H^{\mathcal{T}}(\mathcal{H}; \delta)$ is the structural information of α's and β's subtree δ; $H^{\mathcal{T}_{mg}(\alpha,\beta)}(\mathcal{H}; \alpha)$ is the structural information of subtree α after merging subtree β into α; Similarly, $H^{\mathcal{T}_{mg}(\alpha,\beta)}(\mathcal{H}; \delta)$ is the structure information of the subtree δ after merging subtree β into α. Eq. 10 calculates the change in the related subtree structure entropy before and after running the Merge operator on nodes α and β. If $\Delta_{\mathcal{H}}^{Mg}(T; \alpha, \beta) \leq 0$, then the Merging operator runs successfully, denoted as $Mg(\mathcal{T}; \alpha, \beta) \downarrow$.

The initial one-dimensional encoding tree implies that each node v_i forms an independent community. Next, we calculate the difference in structural entropy before and after executing the Merge operator on any two nodes and select no more than the maximum number of pairs, denoted as Q, from the pairs of nodes whose structural entropy is reduced the most to run the Merge operator. Continue this process until no pair of nodes is found that

Algorithm 1: Optimized 3-dimensional Encoding Tree on Hypergraph.

Input: bipartite graph matrix of the hypergraph: A_G; hyperparameter for parallel Merge Operators: p.
Output: Three-dimensional community division: S_1 and S_2.

calculate $A_{\mathcal{H}}$ with A_G via Eq. 8;
$n_{cur} \leftarrow len(A_{\mathcal{H}})$, $S_1 \leftarrow$ the initial division;
Initial L_{edge} with node pairs in $A_{\mathcal{H}}$, $flag_m \leftarrow False$;
while *not* $flag_m$ **do**
  $Q \leftarrow ceil((n_{cur} - 1) * p)$ via Eq. 11;
  for (u, v) *in* L_{edge} **do**
    $dH(u, v) \leftarrow \Delta^{Mg}_{\mathcal{H}}(T; u, v)$ via Eq. 10;
  `// Node pairs that can be merged`
  $L_{op} \leftarrow (u, v)$ with $dH(u, v) > median(dH > 0)$;
  if $len(L_{op}) > Q$ **then**
    $L_{op} \leftarrow (u, v)$ with top Q dH;
  if $len(L_{op}) = 0$ **then**
    $flag_m \leftarrow True$;
  `// Update the number of communities`
  $n_{cur} \leftarrow n_{cur} - len(L_{op})$;
  `// Running Merge Operator on node pairs in` L_{op}
  update L_{edge} and S_1;
calculate $A_G \leftarrow {S_1}^{\intercal} \cdot A_G$ via Eq. 12; repeat Line 1-14;

allows the Merge operator to execute successfully. The value of Q for each iteration is determined as follows:

$$Q = ceil((n_{cur} - 1) \times p), \tag{11}$$

where n_{cur} represents the current number of communities/nodes, and p is a hyperparameter between 0 and 1 that controls the speed of parallel operations for the Merge operator. The operator *ceil* is a mathematical function that rounds a given number up to the nearest integer. The final community division can be expressed mathematically as S_1. Therefore, by treating the community set as a new set of nodes, we can obtain the bipartite graph matrix A'_G of the community hypergraph:

$$A'_G = {S_1}^{\intercal} \cdot A_G, S_1 = (s_{i,j}) \in \{0, 1\}^{|\mathcal{V}| \times |C_1|}, \tag{12}$$

where C_1 is the preliminary community set, $s_{i,j}$ indicates whether node v_i belongs to community c_1^j, and A_G is the bipartite graph matrix of the initial hypergraph. The A'_G obtained in this way represents the connection relationship of the hyperedges between communities, and there will be self-loops. As outlined in Algo. 1, by repeating the calculation of Eq. 8- 10 and the above two-dimensional encoding tree construction algorithm, the final community set C_2 with a higher-level community division S_2 can be obtained. The construction of the three-dimensional optimal encoding tree and the corresponding community division is completed at this point.

3.3 Hypergraph Pooling

The three-dimensional encoding tree successfully learns the topological structure of the hypergraph, dividing nodes into two hierarchical communities. Building upon this foundation, this subsection introduces a method for

capturing node group characteristics. We introduce pooling and unpooling layers after the hypergraph encoder to enhance the diffusion process of node information.

Pooling and unpooling. The pooling operation aggregates information embedded at low-level nodes to form representations of high-level communities. In contrast, the unpooling operation, conversely, propagates the obtained representations of high-level communities back to low-level nodes. These operations can be mathematically expressed as:

$$\mathbf{E_c} = S^\top \cdot \mathbf{E_v}, \tag{13}$$

$$\mathbf{E_v} = S \cdot \mathbf{E_c}. \tag{14}$$

Eq. 13 is the pooling operation, and Eq. 14 is the unpooling operation, where S is the community division matrix, which is the same as in Subsection 3.2. $\mathbf{E_c}$ and $\mathbf{E_v}$ are the embedding representations of community sets and node sets, respectively.

Encoder-decoder for hypergraph pooling. In the hypergraph pooling encoder (HP), two upward abstraction modules are employed, each composed of an HGCN layer and a pooling layer. In the hypergraph unpooling decoder(HPU), two downward feedback modules are utilized, each consisting of an unpooling layer and an HGCN layer. Another HGCN layer is also inserted between the encoder and decoder to facilitate the diffusion of information of the highest-level communities. To simultaneously capture global community representations and direct neighbor dependencies, the HGCN layer in the decoder does not directly utilize the output of the unpooling layer. Instead, it combines the representations from the HGCN layer output in the corresponding encoder and the unpooling layer output through weighting. This encoder-decoder skip-connection operation enables the embedding of upper-level community information into the embeddings of lower-level nodes. Otherwise, each bottom-level node will ultimately receive feedback that only differs slightly in terms of community embedding. The detailed process of hypergraph pooling is depicted in Algo. 2

Specifically, as illustrated in Figure 3, for the initial hypergraph $\mathcal{H}_0$, we first randomize a $|\mathcal{V}| \times d$ node embedding matrix $\mathbf{E}_\mathbf{v}^0$. Then, after one round of information propagation through the HGCN layer, we obtain an embedding matrix $\mathbf{E}_\mathbf{v}^1$ of the same size. Subsequently, based on the community division matrix S_1 on the encoding tree in Subsection 3.2, we perform embedding pooling to abstract upwards and obtain a preliminary community embedding matrix $\mathbf{E}_\mathbf{v}^2$ of size $|C_1| \times d$. The updated hypergraph after node aggregation is the community hypergraph $\mathcal{H}_1$. After the second round of HGCN information propagation, we obtain an embedding matrix $\mathbf{E}_\mathbf{v}^3$ of size $|C_1| \times d$. Next, based on the community division matrix S_2 in Subsection 3.2, we perform preliminary community pooling to abstract upwards and obtain the final community embedding matrix $\mathbf{E}_\mathbf{v}^4$ of size $|C_2| \times d$. At this point, the work of the encoder is complete. The updated hypergraph after preliminary community aggregation is the final community hypergraph $\mathcal{H}_2$, and after one round of HGCN information propagation, we obtain an embedding matrix $\mathbf{E}_\mathbf{v}^5$ of size $|C_2| \times d$. Then, start the work of the decoder. Firstly, based on the community division matrix S_2, the final community unpooling is performed to provide the preliminary community embedding matrix $\mathbf{E}_\mathbf{v}^6$ of size $|C_1| \times d$. Subsequently, $\mathbf{E}_\mathbf{v}^6$ is weighted with the preliminary community embedding matrix $\beta_{hpu}\mathbf{E}_\mathbf{v}^3$ at the corresponding level in the encoder, input into the HGCN layer, and after one round of propagation, an embedding matrix $\mathbf{E}_\mathbf{v}^7$ of size $|C_1| \times d$ is obtained. Next, based on the community division matrix S_1, the preliminary community unpooling is carried out to provide a node embedding matrix $\mathbf{E}_\mathbf{v}^8$ of size $|V| \times d$. Finally, $\mathbf{E}_\mathbf{v}^8$ is weighted with the node embedding matrix $\beta_{hpu}\mathbf{E}_\mathbf{v}^1$ in the encoder and passed into the HGCN layer to obtain the final node embedding matrix $\mathbf{E}_\mathbf{v}^9$ of size $|\mathcal{V}| \times d$. Essentially, the pooling and unpooling processes described above aim to obtain upper-level community embeddings $\mathbf{E}_\mathbf{v}^8$ of nodes. For the original hypergraph $\mathcal{H}_0$, this is equivalent to performing two rounds of hypergraph encoding: 1) The initial node embeddings $\mathbf{E}_\mathbf{v}^0$, following a round of HGCN propagation, yield new node embeddings $\mathbf{E}_\mathbf{v}^1$ with direct neighbor dependencies; 2) the new node embeddings $\mathbf{E}_\mathbf{v}^1$ are weighted with the corresponding upper-level community embeddings $\mathbf{E}_\mathbf{v}^8$. After the second round of HGCN

Algorithm 2: Hypergraph Pooling and Unpooling via 3-dimensional Encoding Tree.

Input: community divisions: S_1 and S_2; bipartite graph matrix of initial hypergraph: $A^0_{\mathcal{G}}$; initial node embedding: $\mathbf{E}^0_{\mathbf{v}}$, the weight of the previous embedding in HPU: β_{hpu}.

Output: final node embedding: $\mathbf{E}^9_{\mathbf{v}}$.

calculate $A^1_{\mathcal{G}}$ with S_1 and $A^0_{\mathcal{G}}$ via Eq. 12;
calculate $A^2_{\mathcal{G}}$ with S_2 and $A^1_{\mathcal{G}}$ via Eq. 12;
initial $L_E \leftarrow empty$;
for $i = 0, 1$ **do**
    // HGCN
    calculate $\mathbf{E}^{2*\mathbf{i}+1}_{\mathbf{v}}$ with $A^i_{\mathcal{G}}$ and $\mathbf{E}^{2*\mathbf{i}}_{\mathbf{v}}$ via Eq. 1;
    L_E append $\mathbf{E}^{2*\mathbf{i}+1}_{\mathbf{v}}$;
    // HP
    calculate $\mathbf{E}^{2*\mathbf{i}+2}_{\mathbf{v}}$ with S_{i+1} and $\mathbf{E}^{2*\mathbf{i}+1}_{\mathbf{v}}$ via Eq. 13;
for $i = 2, 1$ **do**
    // HGCN
    calculate $\mathbf{E}^{9-2*\mathbf{i}}_{\mathbf{v}}$ with $A^i_{\mathcal{G}}$ and $\mathbf{E}^{8-2*\mathbf{i}}_{\mathbf{v}}$ via Eq. 1;
    // HPU
    calculate $\mathbf{E}^{10-2*\mathbf{i}}_{\mathbf{v}}$ with S_i and $\mathbf{E}^{9-2*\mathbf{i}}_{\mathbf{v}}$ via Eq. 14;
    $\mathbf{E}^{10-2*\mathbf{i}}_{\mathbf{v}} \leftarrow \mathbf{E}^{10-2*\mathbf{i}}_{\mathbf{v}} + \beta_{\mathbf{hpu}} * \mathbf{L_E}[\mathbf{i}-\mathbf{1}]$;
calculate $\mathbf{E}^9_{\mathbf{v}}$ with $A^0_{\mathcal{G}}$ and $\mathbf{E}^8_{\mathbf{v}}$ via Eq. 1;

propagation, the final node embeddings $\mathbf{E}^9_{\mathbf{v}}$ are obtained with both direct neighbor dependencies and global collective representations.

3.4 Recommendation and Optimization

The entire recommendation process in EPRHSE is depicted as Algo. 3. Initially, it involves constructing hypergraphs for both primary and auxiliary tasks. Subsequently, for each hypergraph, the 3-dimensional encoding tree is built using hypergraph structural entropy (Algo. 1) to partition node communities hierarchically. Based on these community node mapping relationships, hypergraph pooling and unpooling (Algo. 2) are performed to enhance the aggregation process of the hypergraph encoder. We conduct pre-training optimization based on different task types. For the main task, i.e., the recommendation task, we compute the ranking score Y_{rec} for the user-item pair using inner product calculation:

$$Y_{rec} = \mathbf{E_u} \cdot \mathbf{E_i}^\top, \tag{15}$$

where $\mathbf{E_u}$ and $\mathbf{E_i}$ represent the user and item embedding matrices outputted by the optimized HGCN layer from main tasks, respectively. Y_{rec} is a score matrix of dimensions $|U| \times |I|$, where each element $y_{rec}(u, i)$ signifies the likelihood of recommendation for user u and item i. We employ alignment loss to optimize the recommendation task:

$$\mathcal{L}_{rec} = \frac{1}{|\mathcal{D}_{rec}|} \sum_{(u,i)\in\mathcal{D}_{rec}} \|\mathbf{e_u} - \mathbf{e_i}\|_2{}^2 + \lambda_\Theta \cdot \|\Theta\|_2{}^2, \tag{16}$$

where $\mathcal{D}_{rec}$ represents the set of user-item interaction pairs in the training set, while $\mathbf{e_u}$ and $\mathbf{e_i}$ denote the embedding vectors for user u and item i respectively. Θ represents all trainable parameters in the EPRHSE,

Algorithm 3: Recommendation and Optimization.

Input: hypergraphs for main and auxiliary tasks: $\{A^{u}_{\mathcal{G}}, A^{i}_{\mathcal{G}}, A^{age}_{\mathcal{G}}, A^{job}_{\mathcal{G}}, A^{cate}_{\mathcal{G}}, A^{rate}_{\mathcal{G}}\}$.
Output: Recommendation sequence.

```
// 3-dimensional community divisions
construct {(S_1^u, S_2^u), (S_1^cate, S_2^cate), (S_1^rate, S_2^rate)} for {A_G^u, A_G^i, A_G^age, A_G^job, A_G^cate, A_G^rate} via Algo. 1;
initial E_u and E_i;
// Pre-training
for epoch = 1, 2, 3, ... do
    // main tasks
    update E_u and E_i via Algo. 2;
    // auxiliary tasks
    update {E_u^age, E_u^job} and {E_i^cate, E_i^rate} via Algo. 2;
    // pre-training loss
    calculate L_pre via Eq. 19;
    L_pre backward;
// Fine-tuning
for epoch = 1, 2, 3, ... do
    update E_u and E_i via Algo. 2;
    calculate L_rec via Eq. 16;
    L_rec backward;
Recommend via Eq. 15;
```

encompassing the initial user and item embeddings, denoted as $\Theta = \{\mathbf{E_u} \cup \mathbf{E_i}\}$. Regularization is carried out using λ_Θ.

For auxiliary tasks, take "items with the same cate" as an example. Since hyperedges represent relations and attributes, we utilize the inner product between the embeddings representing this relation (or attribute) hyperedges and the corresponding item (or user) node embeddings as the prediction score:

$$Y_{cate} = \mathbf{E_i} \cdot \mathbf{E_e^{cate}}^{\top}, \tag{17}$$

where $\mathbf{E_e^{cate}}$ and $\mathbf{E_i}$ represent the embeddings of specific hyperedges for the auxiliary task and the embeddings of task-relevant nodes, respectively. Y_{cate} is a score matrix, where each element $y_{cate}(i, e)$ denotes the likelihood of association between item node i and attribute hyperedge e. Subsequently, we employ the Bayesian Personalized Ranking (BPR) loss to optimize the auxiliary task:

$$\mathcal{L}_{cate} = \sum_{(i,e,e') \in \mathcal{D}_{cate}} -log\sigma(y_{cate}(i, e) - y_{cate}(i, e')), \tag{18}$$

where $\mathcal{D}_{cate}$ represents the set of node-hyperedge interaction pairs in the training set of auxiliary task "items with same cate", and each node i is connected to a positive example attribute hyperedge e and a negative example attribute hyperedge e'. $y_{cate}(i, e)$ and $y_{cate}(i, e')$ denote the prediction scores for the positive and negative example attribute hyperedges with node, respectively. σ refers to the Sigmoid function defined as $\sigma(x) = \frac{1}{1+\exp(-x)}$.

Finally, we jointly optimize the pre-trained using both recommendation main tasks and auxiliary tasks:

$$\mathcal{L}_{pre} = \lambda_{rec} \cdot \mathcal{L}_{rec} + (1 - \lambda_{rec}) \cdot \mathcal{L}_a, \mathcal{L}_a = \mathcal{L}_{cate} + ... + \mathcal{L}_{age}, \tag{19}$$

where λ_{rec} serves as the coefficient to balance the losses between main tasks and auxiliary tasks. $\mathcal{L}_a$ represents the auxiliary loss added by all auxiliary tasks. During fine-tuning, only the main task from pre-training is retained. Specifically, during optimization, only the information on the optimized hypergraph encoder for the user-item hypergraph and item-user hypergraph is aggregated. The loss calculation remains partially the same as in pre-training $\mathcal{L}_{rec}$.

3.5 Time complexity analysis

Hypergraph Structural Entropy. The enhanced HGCN in EPRHSE is extended by hypergraph structure entropy. Constructing one 3-dimensional hypergraph structural entropy encoding tree takes $O(\|A_{\mathcal{H}}\|_0 \cdot (1 + log|\mathcal{V}|))$, where $A_{\mathcal{H}}$ is the initial hypergraph adjacent matrix calculate by Eq. 8 and $\mathcal{V}$ is the node set in hypergraph. $\|\cdot\|_0$ is zero norm, representing the number of non-zero elements. Specifically, the number of Megre attempts per round is $\|A_{\mathcal{H}}\|_0$. When the parallel rate is p, the maximum number of rounds required for merging is $log_{1-p}\frac{1-p}{|\mathcal{V}|-1}$. Consequently, the time complexity of constructing a 2-dimensional encoding tree is $O(\|A_{\mathcal{H}}\|_0 \cdot log_{1-p}\frac{1-p}{|\mathcal{V}|-1})$, which can be simplified to $O(\|A_{\mathcal{H}}\|_0 \cdot (1 + log|\mathcal{V}|))$. Similarly, the time complexity of adding one layer of a 2-dimensional encoding tree is $O(\|A_{\mathcal{H}'}\|_0 \cdot (1 + log|C_1|))$, where C_1 is the set of community nodes in the previous 2-dimensional encoding tree, and $A_{\mathcal{H}'}$ is the adjacency matrix of the hypergraph updated by Eq. 12 and Eq. 8. Thus, the time complexity of constructing a 3-dimensional encoding tree is $O(\|A_{\mathcal{H}}\|_0 \cdot (1 + log|\mathcal{V}|) + \|A_{\mathcal{H}'}\|_0 \cdot (1 + log|C_1|))$, which can be simplified to $O(\|A_{\mathcal{H}}\|_0 \cdot (1 + log|\mathcal{V}|))$. Incorporated into our model EPRHSE, we construct a 3-dimensional encoding tree for tasks of user-item, i-cate, and i-rate. Therefore, the added time complexity is $O(\|A^u_{\mathcal{H}}\|_0 \cdot (1 + log|U|) + (\|A^{cate}_{\mathcal{H}}\|_0 + \|A^{rate}_{\mathcal{H}}\|_0) \cdot (1 + log|I|))$, where $A^u_{\mathcal{H}}, A^{cate}_{\mathcal{H}}$, and $A^{rate}_{\mathcal{H}}$ represent the hypergraph adjacent matrix for three tasks, respectively. U is the user set and I is the item set.

Optimized HGCN and pre-training. The time complexity of a single layer in an HGCN is $O(|\mathcal{V}|^2 \cdot d + nnz(A_G) \cdot |\mathcal{V}|)$, where $|\mathcal{V}|$ denotes the number of nodes in the hypergraph, and A_G represents the bipartite graph matrix of the hypergraph. When A_G is sparse, the time complexity of the HGCN can be approximated as $O(|\mathcal{V}|^2 \cdot d)$. The time complexity of HP or HPU is $O(|C| \cdot |\mathcal{V}| \cdot d)$. Therefore, the time complexity of an optimized HGCN is $O(2 \cdot |\mathcal{V}|^2 \cdot d + 2 \cdot |C_1| \cdot |\mathcal{V}| \cdot d + 2 \cdot |C_1|^2 \cdot d + 2 \cdot |C_1| \cdot |C_2| \cdot d + |C_2|^2 \cdot d)$, where $|C_1|$ and $|C_2|$ are the number of communities of two layers. Since $|C_1|$ and $|C_2|$ are typically much smaller compared to $|\mathcal{V}|$, the time complexity of optimized HGCN can be simplified to $O(2 \cdot |\mathcal{V}|^2 \cdot d)$. Since the hypergraph structural entropy can be precomputed and does not need to be repeatedly involved during pre-training, the overall time complexity only considers the optimized HGCN for all tasks. Thus, it can be approximated as $O(2 \cdot |U|^2 \cdot d) + O(|I|^2 \cdot d) + O(2 \cdot |I|^2 \cdot d) + O(2 \cdot |I|^2 \cdot d) + O(|U|^2 \cdot d) + O(|U|^2 \cdot d)$, representing two main tasks and four auxiliary tasks respectively. After simplifying the constant term, it can be further expressed as $O(|U|^2 \cdot d + |I|^2 \cdot d)$, where $|U|$ is the number of users and $|I|$ is the number of items.

3.6 Space complexity analysis

We next analyze the space complexity of the proposed framework EPRHSE from three perspectives: hypergraph structural entropy (Algo. 1), hypergraph pooling/unpooling (Algo. 2), and the overall pre-training and optimization procedure (Algo. 3).

Hypergraph structural entropy. The adjacency matrix $A_{\mathcal{H}}$ derived from Eq. 8 has $nnz(A_{\mathcal{H}})$ non-zero elements. Storing this sparse structure requires $O(nnz(A_{\mathcal{H}}))$ memory. During the construction of the encoding tree, candidate node pairs L_{op} are generated. In the worst case, the number of candidate pairs is $O(|\mathcal{V}|^2)$; however, by parallel merging with parameter p, only $Q = O((1-p)|\mathcal{V}|)$ pairs are retained in each round, leading to an effective peak memory of $O(nnz(A_{\mathcal{H}}) + Q)$. Thus, the space complexity of structural entropy is $O(nnz(A_{\mathcal{H}}) + |\mathcal{V}|)$.

Hypergraph pooling and unpooling. In Algo. 2, the major memory usage comes from storing node and community embeddings. Each layer requires embeddings of size $|\mathcal{V}| \times d$, $|C_1| \times d$, and $|C_2| \times d$, together with the

bipartite graph matrices $A^0_{\mathcal{G}}, A^1_{\mathcal{G}}, A^2_{\mathcal{G}}$. Since $|C_1|, |C_2| \ll |\mathcal{V}|$, the worst overall space complexity of hypergraph pooling/unpooling is $O(|\mathcal{V}| \cdot d + nnz(A_{\mathcal{G}}))$.

Recommendation and optimization. During pre-training and fine-tuning (Algo. 3), we construct encoding trees and learn embeddings for both main and auxiliary tasks. The total memory required for hypergraph structural entropy is $O(nnz(A^u_{\mathcal{H}}) + nnz(A^{cate}_{\mathcal{H}}) + nnz(A^{rate}_{\mathcal{H}}) + |U| + |I|)$. The memory for hypergraph pooling and unpooling is $O(|U|d + nnz(A^u_{\mathcal{G}})) + O(|I|d + nnz(A^i_{\mathcal{G}})) + O(|U|d + nnz(A^{age}_{\mathcal{G}})) + O(|U|d + nnz(A^{job}_{\mathcal{G}})) + O(|I|d + nnz(A^{cate}_{\mathcal{G}})) + O(|I|d + nnz(A^{rate}_{\mathcal{G}}))$. However, since these tasks are optimized sequentially, only one task's embeddings and adjacency structure need to reside in memory at a time. Therefore, the effective peak memory is dominated by the two main tasks: $O(|U|d + nnz(A^u_{\mathcal{G}}) + |I|d + nnz(A^i_{\mathcal{G}}))$. Thus, the total memory requirement can be expressed as $O((|U| + |I|)d + nnz(A^u_{\mathcal{G}}) + nnz(A^i_{\mathcal{G}}) + nnz(A^u_{\mathcal{H}}) + nnz(A^{cate}_{\mathcal{H}}) + nnz(A^{rate}_{\mathcal{H}}))$, which in the worst case can be simplified to $\underline{O(|U||I| + |U|^2 + |I|^2)}$.

In practice, the memory consumption is dominated by the hypergraph structural entropy computation. Since the community partition of the hypergraph is determined solely by the input graph and does not change across training epochs, it can be precomputed once on the CPU and stored to alleviate GPU memory pressure. During training, the precomputed community structure can be efficiently loaded on the GPU as needed. This strategy significantly reduces GPU memory usage and allows the model to scale to large graphs.

4 EXPERIMENT SETUP

In this section, we present the foundational setup of the experiments, including detailed information about the datasets (Section 4.1), the baselines (Section 4.2), variations of EPRHSE (Section 4.3), as well as the calculation of evaluation metrics (Section 4.4).

4.1 Datasets

We conduct experiments on five real-world datasets: Steam, XMrec-CN, XMrec-MX, XMrec-AU, and XMrec-BR. Steam [63] includes users' transaction records on the Steam online game store. In auxiliary tasks, "*items with same category*" (i-cate), "*items with same rate*" (i-rate), "*users with same age*" (u-age), and "*users with same job*" (u-job) are used. XMarket [7] is a publicly available large-scale dataset obtained from Amazon. We use the data from China, Mexico, Australia, and Brazil for our experiments. In auxiliary tasks for these four datasets, "*items with same category*" (i-cate), "*items with same rate*" (i-rate), and "*items bought together*" (i-bT) are used. In addition, substitute relations between "*items compared together*" (i-cpr) are predicted as homogeneous edges for XMrec-MX. Since the other three datasets do not support this data, there is no such auxiliary task. Beyond auxiliary tasks, we also define two main tasks that directly reflect the essence of recommendation: user–item task ("*users buying the same item*"), which models the similarity between users who share common purchases, thereby capturing collective preference patterns on the user side; item–user task ("*items bought by the same user*"), which models the similarity between items co-consumed by the same user, thereby capturing substitutable or complementary relations on the item side. These two tasks correspond to the "user–item" and "item–user" entries in Table 6. Together, they constitute the fundamental prediction objectives, while auxiliary tasks provide additional heterogeneous social and attribute information. For each user, we select one user-item interaction for pre-training and fine-tuning, and the remaining user-item interactions for user-data fine-tuning testing.

The statistics of the five datasets are shown in Table 2. The columns "#user", "#item", "#u-i edges", and "split" are the number of users, the number of items, the number of user-item interactions, and the number of training and test users in the dataset, respectively. The "#sparsity" column indicates the sparsity of the user-item interaction. The "#auxiliary tasks" column indicates the type of auxiliary tasks used for each dataset. For each auxiliary task, we construct a corresponding hypergraph, where each attribute or relationship is modeled as

a hyperedge. The number reported in this column (“hyperedges number”), therefore, corresponds to the total number of hyperedges in the constructed hypergraph. We only adopt the enhanced HGCN encoder for tasks where community aggregation is relatively obvious, explicitly referring to the "users buying same item" main task and the i-cate, i-rate auxiliary task. However, for i-bT, i-cpr, u-age, and u-job auxiliary tasks where community aggregation is not obvious, we retain the original HGCN encoder, and the specific community aggregation can be seen in Table 6.

Table 2. The statistics of datasets.

Dataset	#user	#item	#u-i edges	#sparsity	#auxiliary tasks	split
Steam	50,292	1,809	65,379	0.00072	i-cate: 20 i-rate: 486 u-age: 5,186 u-job: 11,198	train: 50,292 test: 15,087
XMrec-CN	18,806	5,937	23,065	0.00021	i-cate: 12 i-rate: 9 i-bT: 155	train: 18,806 test: 4,259
XMrec-MX	221,890	35,235	305,569	0.00004	i-cate: 14 i-rate: 9 i-bT: 4221 i-cpr: 2309	train: 221,890 test: 83,679
XMrec-AU	86,975	42,094	213,086	0.00006	i-cate: 14 i-rate: 9 i-bT: 3,339	train: 86,975 test: 126,111
XMrec-BR	25,059	11,327	37,072	0.00013	i-cate: 13 i-rate: 9 i-bT: 1,758	train: 25,059 test: 12,013

4.2 Baselines

To verify the effectiveness of EPRHSE, we compare the performances of graph-based recommendation models with and without using user and item embeddings pre-trained by EPRHSE, including LightGCN, DirectAU, UltraGCN, HGNN, HCCF, and DHCF.

- **LightGCN** [32]. This graph convolution network simplifies the design of GCN with only neighborhood aggregation for the recommendation and weighted all layer embeddings.
- **DirectAU** [84]. This graph encoder improves LightGCN by directly optimizing the alignment and uniformity learning objective.
- **UltraGCN** [62]. This is an ultra-simplification of GCN that skips infinite layers of message passing with a constraint loss to approximate the limit of graph convolutions directly.
- **HGNN** [25]. This hypergraph neural network uses hypergraph convolution operations to encode high-order data correlation.
- **HCCF** [102]. This is a self-supervised hypergraph contrastive collaborative filtering framework to jointly capture local and global collaborative relations.

- **DHCF** [40]. This is a variant of HGCN that introduces a dual-channel learning strategy and jump hypergraph convolution method.

We also compare with pre-training baselines, including GCC, SGL, AttriMask, and UPRTH.

- **GCC** [69]. This pre-trained graph contrast encoding model based on HGCN leverages contrastive learning with random walks to capture graph structure information.
- **SGL** [97]. This self-supervised graph-learning model based on LightGCN drops edges for contrastive learning in pre-training.
- **AttriMask** [34]. This method applies HGCN and adds a linear model to predict masked attributes during pre-training.
- **UPRTH** [108]. This unified pre-training framework uses HGCN to learn node embeddings and designs a transitional attention layer to fuse the results of different tasks.

In addition to pre-training models, we further incorporate state-of-the-art graph contrastive learning baselines. These models leverage various data augmentation strategies to perform contrastive learning, thereby addressing the data sparsity issue in recommendation. Representative methods include NCL, XSimGCL, and LightGCL.

- **NCL** [53]. This method incorporates the neighborhood of users (or items) from both the graph structure and the semantic space, and introduces a novel structural adversarial objective that treats structural neighbors as positive contrastive pairs.
- **XSimGCL** [113]. This extremely simple graph contrastive learning method discards ineffective graph augmentations and instead employs a simple yet effective noise-based embedding perturbation to generate contrastive learning views.
- **LightGCL** [9]. This simple yet effective graph contrastive learning paradigm leverages singular value decomposition for contrastive augmentation, thereby enabling unconstrained structural enhancement through global relational modeling.

4.3 Variations

To demonstrate the effectiveness of some modules and prove that good results can be achieved without the hypergraph fusion mechanism, we construct some variants and compare them with EPRHSE by ablation experiments.

- **EPRHSE+TA**$_{att}$ (in Section 5.2.2). This is a variation of EPRHSE with the TA layer introduced in Section 2.4 to integrate the node representations obtained from auxiliary tasks into the main task through an attention mechanism.
- **EPRHSE+TA**$_{sum}$ (in Section 5.2.2). This is a variation of EPRHSE in which the TA layer is incorporated, where the embeddings from the auxiliary task are directly added to the hyperedges of the main task.
- **EPRHSE+TA**$_{concat}$ (in Section 5.2.2). This is a variation of EPRHSE that incorporates the TA layer, where the embeddings of the auxiliary task are concatenated and merged into the hyperedges of the main task through a linear layer.
- **EPRHSE_w/o_auxiliary** (in Section 5.4). This is a variation of EPRHSE without all auxiliary tasks, including i-cate, i-rate, i-bT, i-cpr, u-age, and u-job.

4.4 Evaluation Metrics

We evaluate the recommendation framework by ranking the test items with all non-interacted users during fine-tuning. $Racall@\{10, 20\}$ and $NDCG@\{10, 20\}$ are adopted as evaluation metrics. In the subsequent tables, we use R@10, R@20, N@10, and N@20 for shorthand. Recall@K represents the proportion of correctly predicted items

within the top K recommended items relative to the total number of items that should have been recommended:

$$Recall@K = \frac{TP@K}{TP@K + FN@K}, \tag{20}$$

where $TP@K$ is True Positive when recommending the first K items, and $FN@K$ is False Negative when recommending the first K items. NDCG is Normalized Discounted Cumulative Gain, which takes into account the order of recommended items:

$$NDCG@K = \frac{DCG@K}{IDCG@K}, \quad DCG@K = \sum_{i=1}^{K} \frac{y_{rec}(,i)}{log_2(i+1)}, \quad IDCG@K = \sum_{i=1}^{min\{K,|Y|\}} \frac{1}{log_2(i+1)}, \tag{21}$$

where Y_{rec} is the score matrix between users and items calculated by Eq. 15, and $y_{rec}(,i)$ is one element in Y_{rec} representing the relevance score of item i for a user. Y is a list of real recommended items for a user. The Discounted Cumulative Gain (DCG) is a metric that increases the gain for items ranked higher in the predicted order while discounting the gain for items ranked lower. The Ideal Discounted Cumulative Gain (IDCG) represents the DCG in the best possible item ranking.

5 RESULTS AND DISCUSSION

This section conducts several experiments to evaluate the performance of EPRHSE. We mainly answer the following questions:

- Q1: (Section 5.1) How does EPRHSE perform compared to 13 baseline models across different datasets? Additionally, what advantages does the pre-training architecture of EPRHSE offer over the latest baseline model in terms of algorithmic effectiveness?
- Q2: (Section 5.2) How does the hypergraph pooling layer contribute to overall performance? Additionally, how does the performance change when incorporating the hypergraph fusion mechanism (TA layer)?
- Q3: (Section 5.3) How do four key hyperparameters (learning rate for pre-training lr_{pre}, learning rate for fine-tuning lr, weight for recommendation λ_{rec}, and regularization λ_{Θ}) impact the performance and stability of EPRHSE?
- Q4: (Section 5.4) How does EPRHSE work in the cold-start scenario compared to pre-trained baselines, and which part plays a key role in EPRHSE?
- Q5: (Section 5.5) How efficient can different baselines and EPRHSE be? And what are the advantages of EPRHSE in operation?
- Q6: (Section 5.6) How do the visualization results demonstrate the hierarchical communities formed through structural entropy? And how do the embeddings learned by different models appear when visualized?

5.1 Overall Effectiveness

Table 3 reports the overall results of EPRHSE and thirteen baselines across five datasets. The first six columns correspond to baselines without pre-training, the next three columns correspond to graph contrastive learning baselines, and the following four columns correspond to pre-training-based baselines. After these, we present the results of EPRHSE. The final column (Improv.) reports the percentage improvement and the p-value from the t-test of our model compared with the best-performing baseline. In addition, the standard deviations (std) are reported in a smaller font beneath each row. From these results, we draw the following conclusions:

(1) Overall, the EPRHSE model, which incorporates the topological structure and global dependency relationships, achieves the best recommendation performance across the five datasets. Specifically, except for NDCG@10 and NDCG@20 on the XMrec-MX dataset, which are suboptimal, all other metrics are optimal and show varying degrees of improvement compared to the best baseline. The Recall@10 on

Table 3. Comparison of the Recall@10, Recall@20, NDCG@10, and NDCG@20 across different methods on five datsets. The optimal results are bolded, and the suboptimal results are underlined.

Method		LightGCN	DirectAU	UltraGCN	HGNN	HCCF	DHCF	NCL	XSimGCL	LightGCL	GCC	SGL	AttriMask	UPRTH	EPRHSE	Improv.
Steam	**R@10**	0.0069	0.0574	0.0416	0.0780	0.0976	0.1214	0.0949	0.1121	0.1155	0.1364	0.1120	0.1398	0.1721	**0.1731**	0.59%
	std	± 0.0001	± 0.0055	± 0.0011	± 0.0012	± 0.0007	± 0.0021	± 0.0010	± 0.0013	± 0.0047	± 0.0013	± 0.0028	± 0.0025	± 0.0012	± 0.0002	p=4.1e-2
	R@20	0.0103	0.0860	0.0787	0.1104	0.1453	0.1959	0.1448	0.1729	0.1775	0.2288	0.1796	0.2247	0.2704	**0.2826**	4.50%
	std	± 0.0001	± 0.0051	± 0.0023	± 0.0017	± 0.0008	± 0.0030	± 0.0017	± 0.0013	± 0.0069	± 0.0012	± 0.0026	± 0.0010	± 0.0010	± 0.0008	p=6.4e-4
	N@10	0.0034	0.0290	0.0206	0.0418	0.0489	0.0566	0.0483	0.0557	0.0575	0.0649	0.0537	0.0714	0.0841	**0.0848**	0.83%
	std	± 0.0001	± 0.0029	± 0.0007	± 0.0006	± 0.0005	± 0.0011	± 0.0003	± 0.0007	± 0.0023	± 0.0007	± 0.0024	± 0.0013	± 0.0007	± 0.0001	p=4.8e-2
	N@20	0.0042	0.0362	0.0299	0.0502	0.0609	0.0753	0.0609	0.0710	0.0731	0.0882	0.0708	0.0929	0.1088	**0.1122**	3.11%
	std	± 0.0001	± 0.0028	± 0.0006	± 0.0008	± 0.0003	± 0.0013	± 0.0004	± 0.0006	± 0.0026	± 0.0007	± 0.0023	± 0.0010	± 0.0006	± 0.0002	p=3.1e-3
XMrec-CN	**R@10**	0.0036	0.0030	0.0031	0.0061	0.0023	0.0033	0.0053	0.0034	0.0046	0.0112	0.0062	0.0551	0.0801	**0.0864**	7.85%
	std	± 0.0004	± 0.0004	± 0.0002	± 0.0010	± 0.0002	± 0.0005	± 0.0010	± 0.0006	± 0.0007	± 0.0007	± 0.0004	± 0.0006	± 0.0015	± 0.0004	p=7.0e-4
	R@20	0.0068	0.0040	0.0053	0.0074	0.0038	0.0047	0.0082	0.0074	0.0061	0.0177	0.0065	0.0784	0.1093	**0.1220**	11.57%
	std	± 0.0008	± 0.0005	± 0.0006	± 0.0009	± 0.0007	± 0.0007	± 0.0014	± 0.0004	± 0.0006	± 0.0008	± 0.0003	± 0.0007	± 0.0018	± 0.0007	p=1.4e-4
	N@10	0.0015	0.0014	0.0017	0.0029	0.0008	0.0022	0.0023	0.0014	0.0021	0.0068	0.0038	0.0269	0.0428	**0.0451**	5.44%
	std	± 0.0001	± 0.0002	± 0.0002	± 0.0004	± 0.0002	± 0.0002	± 0.0004	± 0.0002	± 0.0003	± 0.0004	± 0.0002	± 0.0003	± 0.0008	± 0.0003	p=3.9e-3
	N@20	0.0024	0.0017	0.0023	0.0033	0.0012	0.0026	0.0029	0.0024	0.0025	0.0086	0.0039	0.0336	0.0506	**0.0551**	8.81%
	std	± 0.0002	± 0.0002	± 0.0002	± 0.0004	± 0.0003	± 0.0003	± 0.0005	± 0.0002	± 0.0003	± 0.0003	± 0.0002	± 0.0003	± 0.0008	± 0.0002	p=1.4e-4
XMrec-MX	**R@10**	0.0141	0.0133	0.0034	0.0095	0.0077	0.0341	0.0235	0.0249	0.0189	0.0298	0.0457	0.0500	0.0822	**0.0834**	1.44%
	std	± 0.0003	± 0.0013	± 0.0003	± 0.0006	± 0.0006	± 0.0009	± 0.0019	± 0.0026	± 0.0021	± 0.0053	± 0.0004	± 0.0003	± 0.0003	± 0.0003	p=1.2e-2
	R@20	0.0208	0.0196	0.0059	0.0154	0.0122	0.0489	0.0390	0.0421	0.0274	0.0527	0.0540	0.0748	0.0966	**0.0976**	0.98%
	std	± 0.0002	± 0.0021	± 0.0002	± 0.0008	± 0.0007	± 0.0012	± 0.0019	± 0.0034	± 0.0025	± 0.0065	± 0.0004	± 0.0006	± 0.0013	± 0.0006	p=4.1e-3
	N@10	0.0082	0.0081	0.0017	0.0046	0.0043	0.0211	0.0121	0.0127	0.0105	0.0130	0.0233	0.0261	**0.0574**	0.0502	-
	std	± 0.0005	± 0.0006	± 0.0001	± 0.0004	± 0.0002	± 0.0007	± 0.0007	± 0.0011	± 0.0012	± 0.0037	± 0.0003	± 0.0002	± 0.0004	± 0.0002	
	N@20	0.0100	0.0098	0.0023	0.0062	0.0055	0.0251	0.0160	0.0170	0.0128	0.0187	0.0256	0.0327	**0.0612**	0.0537	-
	std	± 0.0004	± 0.0005	± 0.0002	± 0.0004	± 0.0003	± 0.0008	± 0.0006	± 0.0010	± 0.0013	± 0.0037	± 0.0002	± 0.0002	± 0.0002	± 0.0003	
XMrec-AU	**R@10**	0.0042	0.0028	0.0016	0.0048	0.0034	0.0092	0.0128	0.0131	0.0101	0.0107	0.0069	0.0148	0.0165	**0.0212**	28.78%
	std	± 0.0001	± 0.0002	± 0.0002	± 0.0003	± 0.0004	± 0.0010	± 0.0005	± 0.0004	± 0.0007	± 0.0023	± 0.0003	± 0.0021	± 0.0005	± 0.0007	p=8.1e-3
	R@20	0.0065	0.0051	0.0031	0.0074	0.0056	0.0140	0.0205	0.0207	0.0146	0.0161	0.0105	0.0228	0.0185	**0.0258**	13.07%
	std	± 0.0004	± 0.0002	± 0.0003	± 0.0005	± 0.0003	± 0.0022	± 0.0007	± 0.0005	± 0.0007	± 0.0032	± 0.0008	± 0.0014	± 0.0008	± 0.0009	p=1.1e-2
	N@10	0.0032	0.0021	0.0011	0.0035	0.0023	0.0065	0.0071	0.0068	0.0063	0.0082	0.0049	0.0098	0.0159	**0.0182**	14.06%
	std	± 0.0003	± 0.0001	± 0.0002	± 0.0002	± 0.0006	± 0.0013	± 0.0003	± 0.0002	± 0.0003	± 0.0009	± 0.0003	± 0.0009	± 0.0003	± 0.0003	p=1.3e-2
	N@20	0.0038	0.0027	0.0015	0.0042	0.0030	0.0078	0.0090	0.0087	0.0076	0.0097	0.0059	0.0121	0.0164	**0.0193**	18.08%
	std	± 0.0002	± 0.0001	± 0.0002	± 0.0002	± 0.0002	± 0.0018	± 0.0003	± 0.0002	± 0.0003	± 0.0011	± 0.0007	± 0.0010	± 0.0004	± 0.0003	p=1.2e-2
XMrec-BR	**R@10**	0.0232	0.0257	0.0040	0.0431	0.0782	0.0630	0.0629	0.0435	0.0670	0.0830	0.0843	0.0879	0.1365	**0.1402**	2.72%
	std	± 0.0006	± 0.0026	± 0.0004	± 0.0017	± 0.0005	± 0.0033	± 0.0210	± 0.0193	± 0.0285	± 0.0045	± 0.0012	± 0.0022	± 0.0012	± 0.0006	p=4.9e-3
	R@20	0.0293	0.0390	0.0063	0.0860	0.0874	0.0820	0.0815	0.0917	0.0897	0.0911	0.0955	0.0952	0.1463	**0.1509**	3.17%
	std	± 0.0008	± 0.0015	± 0.0005	± 0.0011	± 0.0014	± 0.0027	± 0.0228	± 0.0197	± 0.0282	± 0.0038	± 0.0009	± 0.0011	± 0.0008	± 0.0002	p=2.1e-3
	N@10	0.0096	0.0106	0.0022	0.0145	0.0373	0.0220	0.0371	0.0174	0.0298	0.0608	0.0651	0.0588	0.0696	**0.0939**	34.76%
	std	± 0.0003	± 0.0009	± 0.0003	± 0.0007	± 0.0006	± 0.0016	± 0.0178	± 0.0069	± 0.0119	± 0.0056	± 0.0021	± 0.0007	± 0.0046	± 0.0028	p=7.8e-3
	N@20	0.0114	0.0141	0.0028	0.0251	0.0398	0.0273	0.0420	0.0297	0.0357	0.0658	0.0681	0.0608	0.0723	**0.0968**	33.83%
	std	± 0.0002	± 0.0004	± 0.0002	± 0.0005	± 0.0005	± 0.0024	± 0.0175	± 0.0063	± 0.0113	± 0.0016	± 0.0014	± 0.0015	± 0.0035	± 0.0026	p=8.1e-3

five datasets is increased by 0.59%, 7.85%, 1.44%, 28.78%, and 2.72%, while the Recall@20 is increased by 4.50%, 11.57%, 0.98%, 13.07%, and 3.17% respectively. The NDCG@10 on four datasets (except XMrec-MX) is increased by 0.83%, 5.44%, 14.06%, and 34.76%, while the NDCG@20 is increased by 3.11%, 8.81%, 18.08%, and 33.83%, respectively. Moreover, all 18 p-values are below 0.05, with 9 of them smaller than 0.005, indicating that the superiority of EPRHSE is statistically significant. We attribute these improvements to the use of hypergraph structural entropy in EPRHSE, which integrates topological structures and global dependency relationships during pre-training, thereby enabling the learned embeddings to better capture

Table 4. Recall results of adding EPRHSE for pre-training on baselines without pre-training. (↑ indicates which method ("+UPRTH" or "+ EPRHSE") achieves greater improvement, while red denotes performance drop after pre-training. Cells with light orange highlight results superior to UPRTH, dark orange highlights results superior to EPRHSE, and the best results are highlighted in bold.)

Dataset	**Steam**		**XMrec-CN**		**XMrec-MX**		**XMrec-AU**		**XMrec-BR**	
Metric	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20
EPRHSE	**0.1731**	**0.2826**	**0.0864**	**0.1220**	**0.0834**	**0.0976**	**0.0212**	0.0258	0.1402	0.1509
UPRTH	0.1721	0.2704	0.0801	0.1093	0.0822	0.0966	0.0165	0.0185	0.1365	0.1463
LightGCN	0.0069	0.0103	0.0036	0.0068	0.0141	0.0208	0.0042	0.0065	0.0232	0.0293
+UPRTH	0.0097	0.0165	0.0353	0.0716	↑0.0593	↑0.0717	0.0192	0.0244	0.1302	0.1380
+ EPRHSE	↑0.0250	↑0.0345	↑0.0656	↑0.1126	0.0582	0.0705	↑0.0195	↑0.0252	↑0.1317	↑0.1400
DirectAU	0.0574	0.0860	0.0030	0.0040	0.0133	0.0196	0.0028	0.0051	0.0257	0.0390
+UPRTH	0.0601	0.0898	0.0614	0.1076	↑0.0550	0.0681	0.0198	0.0247	0.1305	0.1447
+ EPRHSE	↑0.0762	↑0.1086	↑0.0695	↑0.1139	0.0545	↑0.0690	↑0.0207	↑0.0261	↑0.1366	↑0.1475
UltraGCN	0.0416	0.0787	0.0031	0.0053	0.0034	0.0059	0.0016	0.0031	0.0040	0.0063
+UPRTH	0.0702	0.1370	0.0526	0.0791	↑0.0559	↑0.0706	0.0194	0.0213	0.1279	0.1363
+ EPRHSE	↑0.1232	↑0.2106	↑0.0609	↑0.0845	0.0483	0.0596	↑0.0201	↑0.0227	↑0.1303	↑0.1403
HGNN	0.0780	0.1104	0.0061	0.0074	0.0095	0.0154	0.0048	0.0074	0.0431	0.0860
+UPRTH	↑0.1334	↑0.2289	0.0678	0.1078	0.0530	0.0623	0.0188	0.0228	0.0999	0.1462
+ EPRHSE	0.1239	0.2191	↑0.0808	↑0.1129	↑0.0587	↑0.0704	↑0.0207	↑0.0270	↑0.1381	↑0.1483
HCCF	0.0976	0.1453	0.0023	0.0038	0.0077	0.0122	0.0034	0.0056	0.0782	0.0874
+UPRTH	0.1450	0.2460	↑0.0461	↑0.0715	0.0364	0.0515	0.0132	0.0142	0.1307	0.1381
+ EPRHSE	↑0.1483	↑0.2488	0.0391	0.0625	↑0.0580	↑0.0678	↑0.0168	↑0.0194	↑0.1340	↑0.1436
DHCF	0.1214	0.1959	0.0033	0.0047	0.0341	0.0489	0.0092	0.0140	0.0630	0.0820
+UPRTH	0.1472	0.2567	↑0.0744	↑0.1172	0.0334	0.0508	0.0171	0.0193	0.1104	0.1226
+ EPRHSE	↑0.1652	↑0.2676	0.0697	0.1134	↑0.0438	↑0.0562	↑0.0190	↑0.0241	↑0.1205	↑0.1363
NCL	0.0949	0.1448	0.0053	0.0082	0.0235	0.0390	0.0128	0.0205	0.0629	0.0815
+UPRTH	0.0725	0.1163	0.0285	0.0409	↑0.0457	↑0.0605	0.0171	0.0243	0.0631	0.0708
+ EPRHSE	↑0.1194	↑0.1976	↑0.0297	↑0.0453	0.0448	0.0594	↑0.0206	**↑0.0292**	↑0.0677	↑0.1287
XSimGCL	0.1121	0.1729	0.0034	0.0074	0.0249	0.0421	0.0131	0.0207	0.0435	0.0917
+UPRTH	0.0706	0.1122	0.0363	0.0458	0.0449	0.0554	0.0165	0.0217	↑0.0923	↑0.0993
+ EPRHSE	0.1023	0.1520	↑0.0384	↑0.0546	↑0.0492	↑0.0643	↑0.0186	↑0.0252	0.0590	0.0677
LightGCL	0.1155	0.1775	0.0046	0.0061	0.0189	0.0274	0.0101	0.0146	0.0670	0.0897
+UPRTH	0.0884	0.1140	0.0278	0.0549	0.0472	0.0599	0.0128	0.0184	**↑0.1714**	**↑0.1838**
+ EPRHSE	↑0.1196	↑0.1986	↑0.0357	↑0.0597	↑0.0511	↑0.0611	↑0.0141	↑0.0198	0.1649	0.1745

the characteristics of users and items. Furthermore, in Section 5.2.1, we conduct an ablation study on the hypergraph pooling layers, which confirms that an appropriate number of hypergraph structural entropy pooling layers contributes to enhanced recommendation performance. This indicates that enhancing the performance of recommendations from the perspective of hypergraph learning and model architecture is effective.

(2) Among the baselines, models enhanced with pre-training or graph contrastive learning (GCL) generally outperform the basic models. Within the subset of pre-trained baselines, those employing auxiliary tasks

Table 5. NDCG results of adding EPRHSE for pre-training on baselines without pre-training. (↑ indicates which method ("+UPRTH" or "+ EPRHSE") achieves greater improvement, while red denotes performance drop after pre-training. Cells with light orange highlight results superior to UPRTH, dark orange highlights results superior to EPRHSE, and the best results are highlighted in bold.)

Dataset	**Steam**		**XMrec-CN**		**XMrec-MX**		**XMrec-AU**		**XMrec-BR**	
Metric	N@10	N@20	N@10	N@20	N@10	N@20	N@10	N@20	N@10	N@20
EPRHSE	**0.0848**	**0.1122**	0.0451	0.0551	0.0502	0.0537	0.0182	0.0193	0.0939	0.0968
UPRTH	0.0841	0.1088	0.0428	0.0506	**0.0574**	**0.0612**	0.0159	0.0164	0.0696	0.0723
LightGCN	0.0034	0.0042	0.0015	0.0024	0.0082	0.0100	0.0032	0.0038	0.0096	0.0114
+UPRTH	0.0047	0.0064	0.0263	0.0371	0.0315	0.0347	0.0174	0.0187	0.0736	0.0757
+EPRHSE	↑0.0127	↑0.0151	↑0.0380	↑0.0514	↑0.0439	↑0.0472	↑0.0177	↑0.0192	↑0.1181	↑0.1203
DirectAU	0.0290	0.0362	0.0014	0.0017	0.0081	0.0098	0.0021	0.0027	0.0106	0.0141
+UPRTH	0.0254	0.0329	0.0349	0.0480	0.0259	0.0295	0.0178	0.0190	↑0.0827	↑0.0865
+EPRHSE	↑0.0359	↑0.0441	↑0.0404	↑0.0527	↑0.0365	↑0.0404	↑0.0183	**↑0.0197**	0.0676	0.0706
UltraGCN	0.0206	0.0299	0.0017	0.0023	0.0017	0.0023	0.0011	0.0015	0.0022	0.0028
+UPRTH	0.0291	0.0459	0.0344	0.0427	↑0.0376	↑0.0415	0.0185	0.0188	↑0.0846	↑0.0867
+EPRHSE	↑0.0582	↑0.0802	↑0.0405	↑0.0476	0.0325	0.0356	**↑0.0187**	↑0.0192	0.0748	0.0773
HGNN	0.0418	0.0502	0.0029	0.0033	0.0046	0.0062	0.0035	0.0042	0.0145	0.0251
+UPRTH	↑0.0648	↑0.0888	0.0368	0.0479	0.0259	0.0284	0.0171	0.0181	0.0594	0.0713
+EPRHSE	0.0586	0.0825	**↑0.0497**	**↑0.0588**	↑0.0269	↑0.0300	↑0.0180	**↑0.0197**	**↑0.1192**	**↑0.1220**
HCCF	0.0489	0.0609	0.0008	0.0012	0.0043	0.0055	0.0023	0.0030	0.0373	0.0398
+UPRTH	0.0715	0.0968	↑0.0277	↑0.0351	0.0233	0.0279	0.0129	0.0130	↑0.0843	↑0.0863
+EPRHSE	↑0.0728	↑0.0980	0.0246	0.0311	↑0.0335	↑0.0361	↑0.0153	↑0.0159	0.0714	0.0740
DHCF	0.0566	0.0753	0.0022	0.0026	0.0211	0.0251	0.0065	0.0078	0.0220	0.0273
+UPRTH	0.0736	0.1010	↑0.0458	↑0.0580	0.0167	0.0213	0.0163	0.0167	0.0514	0.0548
+EPRHSE	↑0.0805	↑0.1061	0.0398	0.0524	↑0.0227	↑0.0260	↑0.0171	↑0.0185	↑0.0601	↑0.0643
NCL	0.0483	0.0609	0.0023	0.0029	0.0121	0.0160	0.0071	0.0090	0.0371	0.0420
+UPRTH	0.0380	0.0491	↑0.0173	↑0.0209	0.0254	0.0293	0.0140	0.0160	↑0.0467	↑0.0489
+EPRHSE	↑0.0572	↑0.0769	0.0162	0.0204	↑0.0298	↑0.0336	↑0.0155	↑0.0179	0.0432	0.0588
XSimGCL	0.0557	0.0710	0.0014	0.0024	0.0127	0.0170	0.0068	0.0087	0.0174	0.0297
+UPRTH	0.0368	0.0473	0.0203	0.0231	0.0258	0.0286	↑0.0144	0.0157	↑0.0663	↑0.0682
+EPRHSE	0.0505	0.0631	↑0.0214	↑0.0261	↑0.0291	↑0.0329	0.0143	↑0.0160	0.0423	0.0446
LightGCL	0.0575	0.0731	0.0021	0.0025	0.0105	0.0128	0.0063	0.0076	0.0298	0.0357
+UPRTH	0.0378	0.0441	0.0159	0.0236	0.0269	0.0302	0.0087	0.0103	0.1088	0.1122
+EPRHSE	0.0568	↑0.0767	↑0.0174	↑0.0237	↑0.0314	↑0.0340	↑0.0099	↑0.0115	↑0.1129	↑0.1172

such as GCC, AttriMask, and UPRTH achieve better results than SGL, which does not incorporate auxiliary tasks. Among the basic models, those using HGCN as the encoder (HGNN, HCCF, and DHCF) perform better than those relying solely on LightGCN (LightGCN and DirectAU). This highlights the importance of pre-training with auxiliary tasks and employing graph contrastive learning for recommendation systems under sparse user–item interactions, as they help augment data under limited knowledge. Furthermore, our EPRHSE improves HGCN by incorporating hypergraph structural entropy, thus overcoming its inability to capture global dependencies and yielding better recommendation performance.

(3) EPRHSE achieves greater improvement in Recall@20 and NDCG@20 compared to Recall@10 and NDCG@10, with the effect being particularly pronounced on the Steam and XMrec-CN datasets. This indicates that EPRHSE is especially effective in enhancing the relevance of items ranked lower in the recommendation list. Consequently, it contributes to recommendation stability, ensuring that relevant suggestions extend beyond just the top-ranked items.

Comparing a pre-trained model with a non-pre-trained model may be inherently unfair. To ensure a fair comparison, we apply EPRHSE's pre-training to the six basic baseline models and three GCL models, using the pre-trained embeddings from EPRHSE as their initial embeddings. Additionally, to validate the effectiveness of our model's pre-training architecture, we conduct the same procedure on the best-performing baseline (UPRTH). The results are presented in Table 4 and Table 5, where the first row of each group represents the original baseline, the second row "+UPRTH" indicates the baseline pre-trained with UPRTH, and the third row "+EPRHSE" represents the baseline pre-trained with EPRHSE. For ease of comparison, we also report the results of EPRHSE and UPRTH, highlighted with a gray background. In the table, ↑ indicates the method ("+UPRTH" or "+EPRHSE") that achieves the larger performance improvement within each group, while a red background indicates that pre-training leads to performance degradation. Cells with a light orange background mark results superior to UPRTH, and those with a dark orange background highlight results superior to EPRHSE. The best results are further emphasized in bold. Both tables demonstrate that the pre-trained embeddings generated by EPRHSE substantially enhance the performance of almost all models, with improvements that are more significant and consistent than those achieved by the best baseline, UPRTH. The only exceptions occur in the steam dataset experiments with XSimGCL and LightGCL, where EPRHSE does not provide performance gains. In contrast, UPRTH exhibits performance degradation across four baselines-DHCF, NCL, XSimGCL, and LightGCL. Moreover, after incorporating EPRHSE for pre-training, many models even surpass the best baseline UPRTH, and in some cases, exceed EPRHSE itself. For example, with EPRHSE pre-training, HGNN achieves higher NDCG scores than EPRHSE on the XMrec-CN, XMrec-AU, and XMrec-BR datasets, establishing new state-of-the-art results. These findings highlight the strong generalization capacity of EPRHSE, enabling it to adapt to diverse downstream models and to deliver remarkable performance advantages in sparse-data scenarios.

5.2 Ablation Study

In this section, we perform ablation experiments on the hypergraph pooling layer (Section 5.2.1), which corresponds to the dimensionality of encoding trees, and examine the impact of incorporating the TA layer in EPRHSE (Section 5.2.2). These experiments aim to validate the contribution and effectiveness of each component.

5.2.1 ***Hypergraph pooling layer.*** To validate that the number of layers in the Hypergraph Pooling of EPRHSE is appropriate, we employ the approach outlined in Section 3.2 to construct higher-dimensional encoding trees and augment the corresponding pooling and unpooling layers in EPRHSE. The experimental results, as depicted in Figure 4 and Figure 5, where each subplot with a dual y-axis illustrates the recommendation performance at the top 10 or 20 across different layers under a dataset. In Figure 4, the blue bar chart corresponding to the left y-axis represents Recall@20 and the red line chart corresponding to the right y-axis represents Recall@10. In Figure 5, the purple bar chart corresponding to the left y-axis represents NDCG@20, and the red line chart corresponding

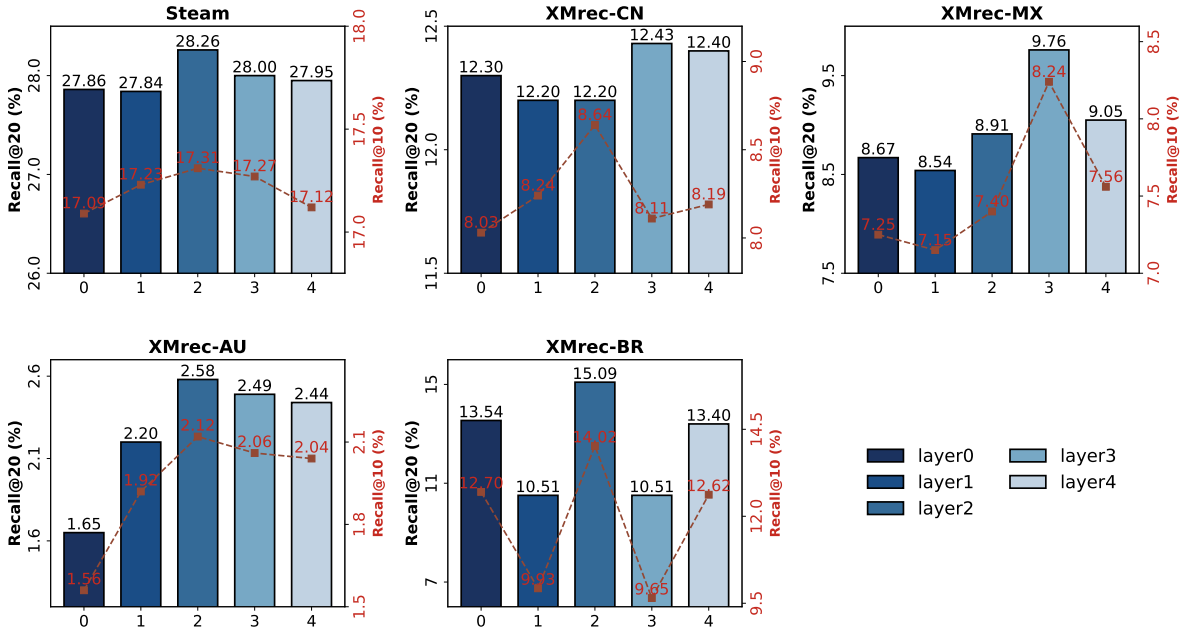


Fig. 4. Recall performance of EPRHSE with Hypergraph Pooling Layer from 0 to 4 layers.

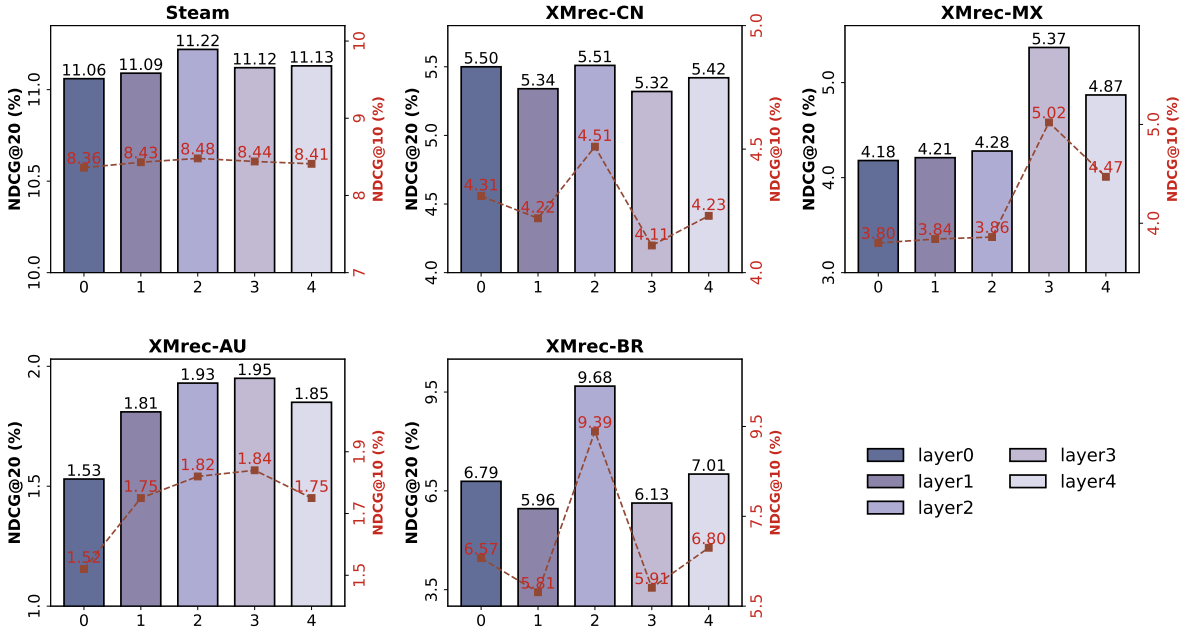


Fig. 5. NDCG performance of EPRHSE with Hypergraph Pooling Layer from 0 to 4 layers.

Table 6. Number of communities on different layers of the encoding tree for five datasets.

Dataset	task type	task name	Layer0	Layer1	Layer2&3&4
Steam	main	user-item	50,292	1,726	1,726
		item-user	1,809	1,809	1,809
	auxiliary	i-cate	1,809	387	381
		i-rate	1,809	666	658
		u-age	50,292	47,159	47,156
		u-job	50,292	23,292	23,030
XMrec-CN	main	user-item	18,806	5,413	5,413
		item-user	5,937	5,937	5,937
	auxiliary	i-cate	5,937	13	12
		i-rate	5,937	10	9
		i-bT	5,937	5,802	5,802
XMrec-MX	main	user-item	221,890	31,222	31,222
		item-user	35,235	35,235	35,235
	auxiliary	i-cate	35,235	14	14
		i-rate	35,235	10	9
		i-bT	35,235	31,425	31,417
		i-cpr	35,235	32,879	32,867
XMrec-AU	main	user-item	86,975	28,090	28,090
		item-user	42,094	42,094	42,094
	auxiliary	i-cate	42,094	15	14
		i-rate	42,094	10	9
		i-bT	42,094	39,414	39,409
XMrec-BR	main	user-item	25,059	7,837	7,837
		item-user	11,327	11,327	11,327
	auxiliary	i-cate	11,327	14	13
		i-rate	11,327	10	9
		i-bT	11,327	9,904	9,899

to the right y-axis represents NDCG@10. For different levels, for example, "*Layeri*" indicates that EPRHSE constructs a (i+1)-dimensional encoding tree to form an i-layer community for pooling and unpooling.

We observe that as the number of layers increases, both Recall@{10,20} and NDCG@{10,20} initially improve and then deteriorate. Except for Recall@20 on the XMrec-CN dataset, NDCG@10, NDCG@20 on the XMrec-AU dataset, and Recall@10, Recall@20, NDCG@10, and NDCG@20 on the XMrec-MX dataset peaking at *Layer*3, all other metrics across the datasets peak at *Layer*2, which is the optimal configuration for the model discussed in EPRHSE. This phenomenon may be related to the number of communities in the encoding tree. Therefore, in Table 6, we report the number of nodes for each task in each dataset, as well as the number of communities obtained after several layers of hypergraph structural entropy partitioning. As seen in Table 6, there is a significant aggregation of nodes from *Layer*0 to *Layer*1, whereas from *Layer*1 to *Layer*2, only a small portion of nodes aggregate to form communities. Even at *Layer*3 and *Layer*4, no new communities are formed. This indicates that a three-layer encoding tree (*Layer*2) is sufficient to learn an adequate hierarchical structure for capturing the global structure. This phenomenon aligns with the results shown in Figure 4 and Figure 5. During the transition from *Layer*0 to *Layer*2, the hierarchical global structure is utilized to update the embeddings of items and users,

significantly improving recommendation performance. However, *Layer*3 and *Layer*4 do not uncover deeper global dependencies, and the repeated pooling and unpooling processes from *Layer*2 result in overfitting of the embeddings. Consequently, the recommendation performance deteriorates.

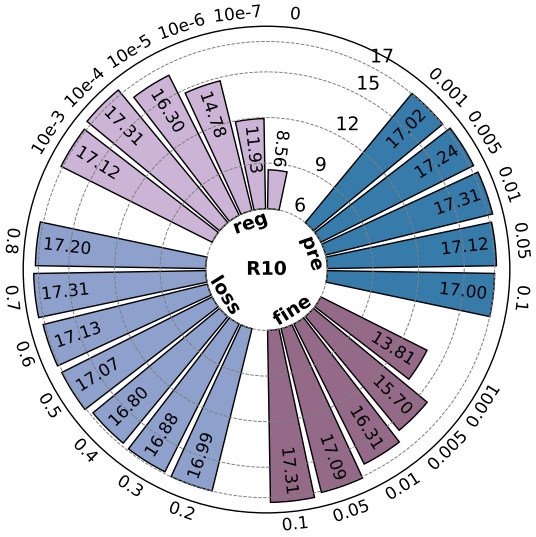
Table 7. Recall@10, Recall@20, NDCG@10, and NDCG@20 results of comparison on TA Layer. The best results are bolded, and the second-best results are underlined.

Dataset	**Steam**		**XMrec-CN**		**XMrec-MX**		**XMrec-AU**		**XMrec-BR**	
Metric	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20
UPRTH	0.1721	0.2704	0.0801	0.1093	0.0822	0.0966	0.0165	0.0185	0.1365	0.1463
EPRHSE+TA$_{att}$	<u>0.1727</u>	<u>0.2815</u>	0.0844	0.1175	0.0824	<u>0.0967</u>	**0.0212**	**0.0258**	0.1392	0.1483
EPRHSE+TA$_{sum}$	0.1719	0.2781	<u>0.0846</u>	<u>0.1178</u>	<u>0.0832</u>	0.0957	**0.0212**	**0.0258**	0.1390	<u>0.1490</u>
EPRHSE+TA$_{concat}$	0.1723	0.2797	0.0802	0.1131	0.0826	0.0961	**0.0212**	**0.0258**	<u>0.1393</u>	0.1481
EPRHSE	**0.1731**	**0.2826**	**0.0864**	**0.1220**	**0.0834**	**0.0976**	**0.0212**	**0.0258**	**0.1402**	**0.1509**
Improv.	0.23%	0.39%	2.12%	3.56%	0.24%	0.93%	-	-	0.64%	1.27%
Metric	N@10	N@20	N@10	N@20	N@10	N@20	N@10	N@20	N@10	N@20
UPRTH	0.0841	0.1088	0.0428	0.0506	**0.0574**	**0.0612**	0.0159	0.0164	0.0696	0.0723
EPRHSE+TA$_{att}$	**0.0849**	<u>0.1118</u>	<u>0.0467</u>	<u>0.0563</u>	0.0477	0.0515	**0.0182**	**0.0193**	0.0886	0.0909
EPRHSE+TA$_{sum}$	0.0841	0.1107	**0.0470**	**0.0565**	0.0481	0.0515	**0.0182**	**0.0193**	0.0910	0.0936
EPRHSE+TA$_{concat}$	0.0845	0.1114	0.0424	0.0520	0.0500	<u>0.0537</u>	**0.0182**	**0.0193**	**0.1018**	**0.1042**
EPRHSE	<u>0.0848</u>	**0.1122**	0.0451	0.0551	<u>0.0502</u>	<u>0.0537</u>	**0.0182**	**0.0193**	<u>0.0939</u>	<u>0.0968</u>
Improv.	-	0.35%	-	-	-	-	-	-	-	-

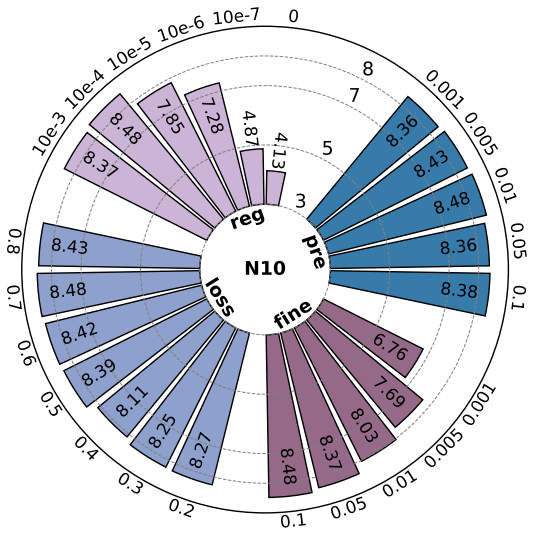
5.2.2 ***TA Layer***. We compare EPRHSE with models that aggregate differently in the TA Layer as well as the best baseline model UPRTH. Three aggregation variations are denoted as EPRHSE+TA$_{att}$, EPRHSE+TA$_{sum}$, and EPRHSE+TA$_{concat}$, respectively. As shown in Table 7, the experimental results indicate that EPRHSE without hypergraph fusion mechanism generally performs best, achieving optimal recommendation results in 13 out of 20 metrics and achieving optimal or suboptimal results in 18 out of 20 scenarios. Moreover, in most cases, employing various aggregation methods, including not aggregating, has improved Recall@{10,20} and NDCG@{10,20} over the best baseline UPRTH, and the differences with EPRHSE are not significant. Even EPRHSE+TA$_{att}$ achieves optimal or suboptimal results in 11 out of 20 scenarios, EPRHSE+TA$_{sum}$ achieves optimal or suboptimal results in 10 out of 20 scenarios, and EPRHSE+TA$_{concat}$ achieves optimal or suboptimal results in 8 out of 20 scenarios. We speculate that this may be because EPRHSE leverages topological structures and global dependencies within its graph encoder to some extent, compensating for the impact of different aggregation methods and thereby reducing the importance of the hypergraph fusion mechanism. Given the limited performance improvement the TA layer provides and its added complexity to the model, we consider not using the hypergraph fusion mechanism in EPRHSE.

5.3 Hyperparameter Sensitivity

In this section, we conduct a sensitivity analysis on four key hyperparameters of EPRHSE: the pre-training learning rate (pre_lr), fine-tuning learning rate (lr), the weight of the recommendation task in the aggregation loss (λ_{rec}), and the regularization weight (λ_Θ). Figure 6 to Figure 10 present the results of EPRHSE on five datasets under different hyperparameter settings. The left-side plots display Recall@10, while the right-side plots show

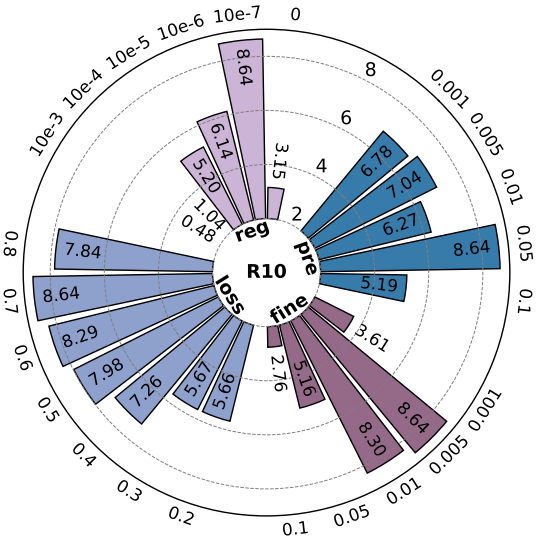


(a) Steam Recall@10

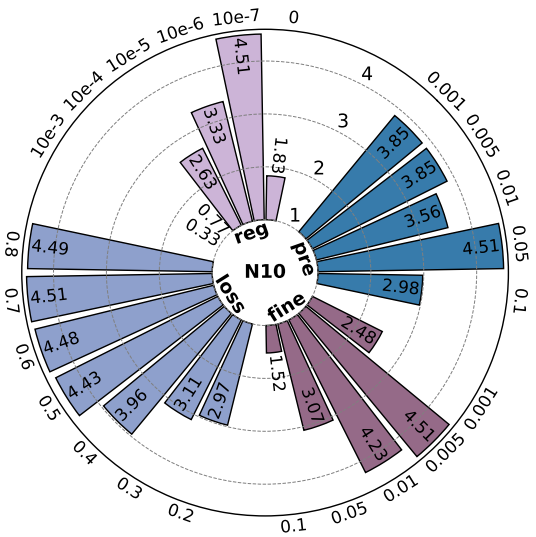


(b) Steam NDCG@10

Fig. 6. Hyperparameter analysis on the Steam dataset ('**pre**' represents the learning rate for pre-training pre_lr, '**fine**' represents the learning rate for fine-tuning lr, '**loss**' represents the weight for recommendation task in loss λ_{rec}, and '**reg**' represents the weight for regularization λ_Θ).

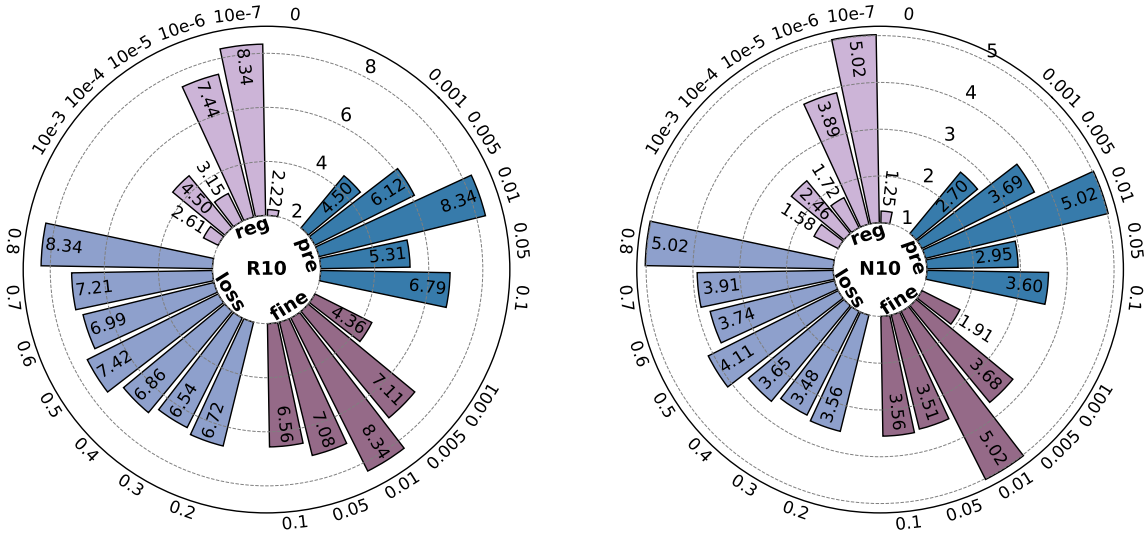


(a) XMrec-CN Recall@10



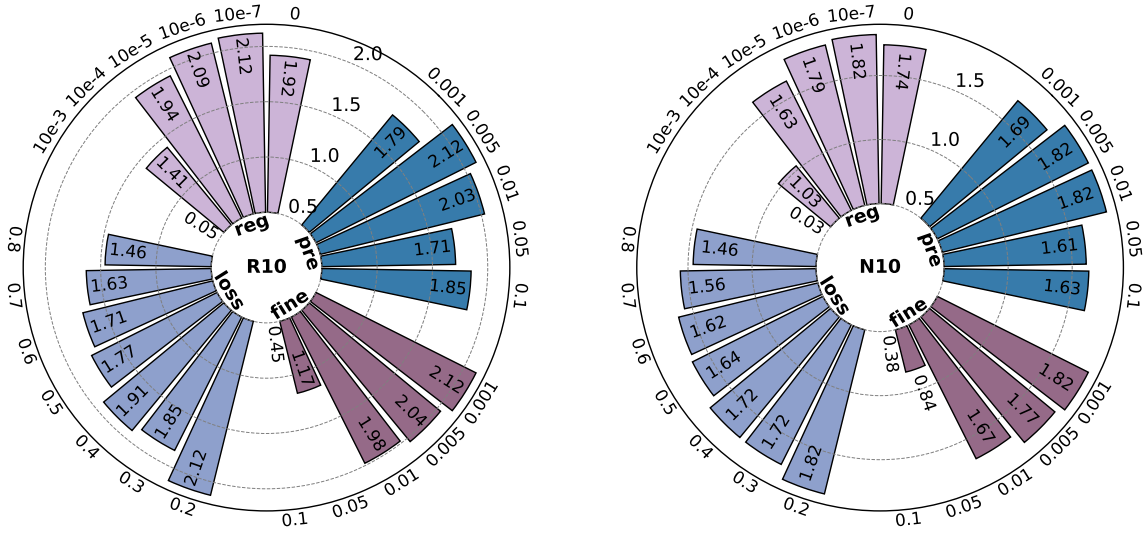
(b) XMrec-CN NDCG@10

Fig. 7. Hyperparameter analysis on the XMrec-CN dataset ('**pre**' represents the learning rate for pre-training pre_lr, '**fine**' represents the learning rate for fine-tuning lr, '**loss**' represents the weight for recommendation task in loss λ_{rec}, and '**reg**' represents the weight for regularization λ_Θ).



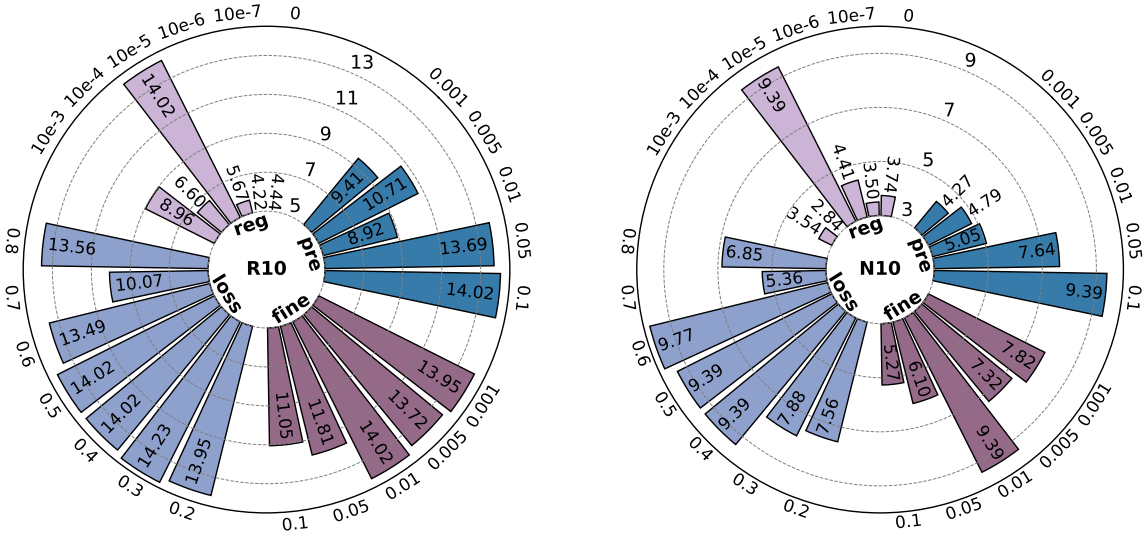
(a) XMrec-MX Recall@10 (b) XMrec-MX NDCG@10

Fig. 8. Hyperparameter analysis on the XMrec-MX dataset ('**pre**' represents the learning rate for pre-training pre_lr, '**fine**' represents the learning rate for fine-tuning lr, '**loss**' represents the weight for recommendation task in loss λ_{rec}, and '**reg**' represents the weight for regularization λ_Θ).



(a) XMrec-AU Recall@10 (b) XMrec-AU NDCG@10

Fig. 9. Hyperparameter analysis on the XMrec-AU dataset ('**pre**' represents the learning rate for pre-training pre_lr, '**fine**' represents the learning rate for fine-tuning lr, '**loss**' represents the weight for recommendation task in loss λ_{rec}, and '**reg**' represents the weight for regularization λ_Θ).



(a) XMrec-BR Recall@10 (b) XMrec-BR NDCG@10

Fig. 10. Hyperparameter analysis on the XMrec-BR dataset ('**pre**' represents the learning rate for pre-training *pre_lr*, '**fine**' represents the learning rate for fine-tuning *lr*, '**loss**' represents the weight for recommendation task in loss λ_{rec}, and '**reg**' represents the weight for regularization λ_Θ).

NDCG@10. Each circular bar chart includes four colors representing the adjustment of a single hyperparameter while keeping others constant. The outer ring values indicate the hyperparameter settings and the inner bars' numerical values represent the corresponding test results.

5.3.1 ***Learning rate for pre-training*** *pre_lr.* For the pre-training learning rate *pre_lr*, we set the selection range to { 0.001, 0.005, 0.01, 0.05, 0.1 } and conduct repeated experiments. As shown in Figure 6 to Figure 10, the optimal choice of *pre_lr* is dataset-dependent. pre_lr has minimal impact on the Steam dataset, whereas an optimal value or range exists for the other four datasets. For example, on the XMrec-CN dataset, 0.05 is the best choice for both Recall and NDCG, with other values leading to a sharp decline in recommendation performance. In the XMrec-MX dataset, 0.01 is the optimal choice since larger and smaller values significantly degrade performance. In the XMrec-AU dataset, 0.005 is the best choice. However, NDCG is more robust to changes in *pre_lr* than Recall, exhibiting stable performance within the range of 0.001 to 0.01. Similarly, on the XMrec-BR dataset, 0.1 is the optimal choice, but Recall is more robust to variations in pre_lr than NDCG, performing well at both 0.05 and 0.1.

5.3.2 ***Learning rate for fine-tuning*** *lr.* For the fine-tuning learning rate *lr*, we set the selection range to { 0.001, 0.005, 0.01, 0.05, 0.1 } and conduct repeated experiments. Selecting an intermediate learning rate is generally beneficial for EPRHSE's recommendation performance; however, the optimal choice remains dataset-dependent. In Figure 6, the Steam dataset exhibits a positive correlation between *lr* and recommendation performance, where a higher learning rate leads to better performance, although the overall fluctuation is relatively small. In Figure 7, the XMrec-CN dataset shows stable performance at $lr = 0.01$ and $lr = 0.005$, while performance drops sharply under other settings. In Figure 8, the XMrec-MX dataset achieves its best performance at $lr = 0.01$. In Figure 9 and Figure 10, the XMrec-AU and XMrec-BR datasets exhibit a negative correlation between *lr* and recommendation

performance, with lower learning rates leading to better results. However, for the XMrec-BR dataset, NDCG@10 performs particularly well at $lr = 0.01$.

5.3.3 ***Weight for recommendation task*** λ_{rec}. For the weight for the recommendation task in loss λ_{rec}, we set the selection range to { 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 } and conduct repeated experiments. The recommendation performance remained relatively stable and favorable during the pre-training phase across most main task loss weight ranges. In particular, the Steam dataset is almost unaffected by variations in this weight. For the XMrec-CN and XMrec-MX datasets, while smaller weight allocations lead to a decline in recommendation performance, performance stabilizes and remains strong when the weight exceeds approximately 0.5. Conversely, for the XMrec-AU and XMrec-BR datasets, larger weight allocations—especially those exceeding 0.6—result in a sharp decline in recommendation performance. However, within the remaining range, performance remains relatively strong.

5.3.4 ***Weight for regularization*** λ_Θ. For the weight for regularization λ_Θ, we set the selection range to { 10e-3, 10e-4, 10e-5, 10e-6, 10e-7, 0 } and conduct repeated experiments. We found that the choice of regularization weight has a unique and significant impact on different datasets, with each dataset exhibiting a distinct "preferred" value. For the Steam dataset in Figure 6, 10e-4 is the optimal choice. Smaller values lead to a sharp decline in performance, while slightly larger values cause a minor performance drop. For the XMrec-CN, XMrec-MX, and XMrec-AU datasets in Figures 7 to Figure 9, 10e-7 is the optimal choice. Despite being an extremely small weight, it remains crucial. Setting it to zero resulted in a substantial decline in recommendation performance. For the XMrec-BR dataset in Figure 10, 10e-5 is the best choice, as both slightly larger and smaller values lead to a near-complete loss of recommendation performance.

5.4 Cold-start

EPRHSE incorporates auxiliary tasks involving attributes and relations during pre-training, enabling it to address the cold-start problem. To demonstrate this, we conceal all relevant user-item interactions for varying proportions of users and then rerun our model. We compare the results of EPRHSE with a variant that is without auxiliary tasks named EPRHSE_w/o_auxiliary, and four pre-trained baselines (GCC, SGL, AttriMask, UPRTH). The experimental results are presented in Figure 11, where the red line represents EPRHSE. Each row corresponds to the results for a specific dataset, with the four columns from left to right representing Recall@10, Recall@20, NDCG@10, and NDCG@20. We conducted experiments using five selected points { 0.1 0.2, 0.3, 0.4, 0.5 }, which indicate the proportion of users without historical interaction data. This proportion can also be interpreted as the fraction of isolated nodes. We find that: (1) Overall, EPRHSE (red line) maintains strong performance in the cold-start scenario, which we attribute to the auxiliary tasks. Notably, on the XMrec-AU dataset, as the proportion of cold-start users increases, Recall@10,20 and NDCG@10,20 exhibit a clear upward trend, even surpassing the performance in scenarios where user-item interaction information is not masked. On the XMrec-BR dataset, EPRHSE does not significantly outperform the best baseline UPRTH, especially when the cold-start user ratio is 20%. However, its overall performance remains stable and is not heavily affected by changes in the inductive ratio. When all auxiliary tasks are disabled (green line), the recommendation performance of EPRHSE_w/o_auxiliary drops sharply compared to both EPRHSE and UPRTH, yet it remains comparable to the other baselines. (2) For the Steam dataset, both EPRHSE and its variant EPRHSE_w/o_auxiliary are mainly unaffected by the proportion of cold-start users and achieve strong recommendation performance. In contrast, AttriMask and UPRTH exhibit a slight downward trend, albeit with minimal fluctuation. This suggests that incorporating the hypergraph structural entropy pooling module provides more excellent benefits for improving performance on the Steam dataset than adding auxiliary tasks.

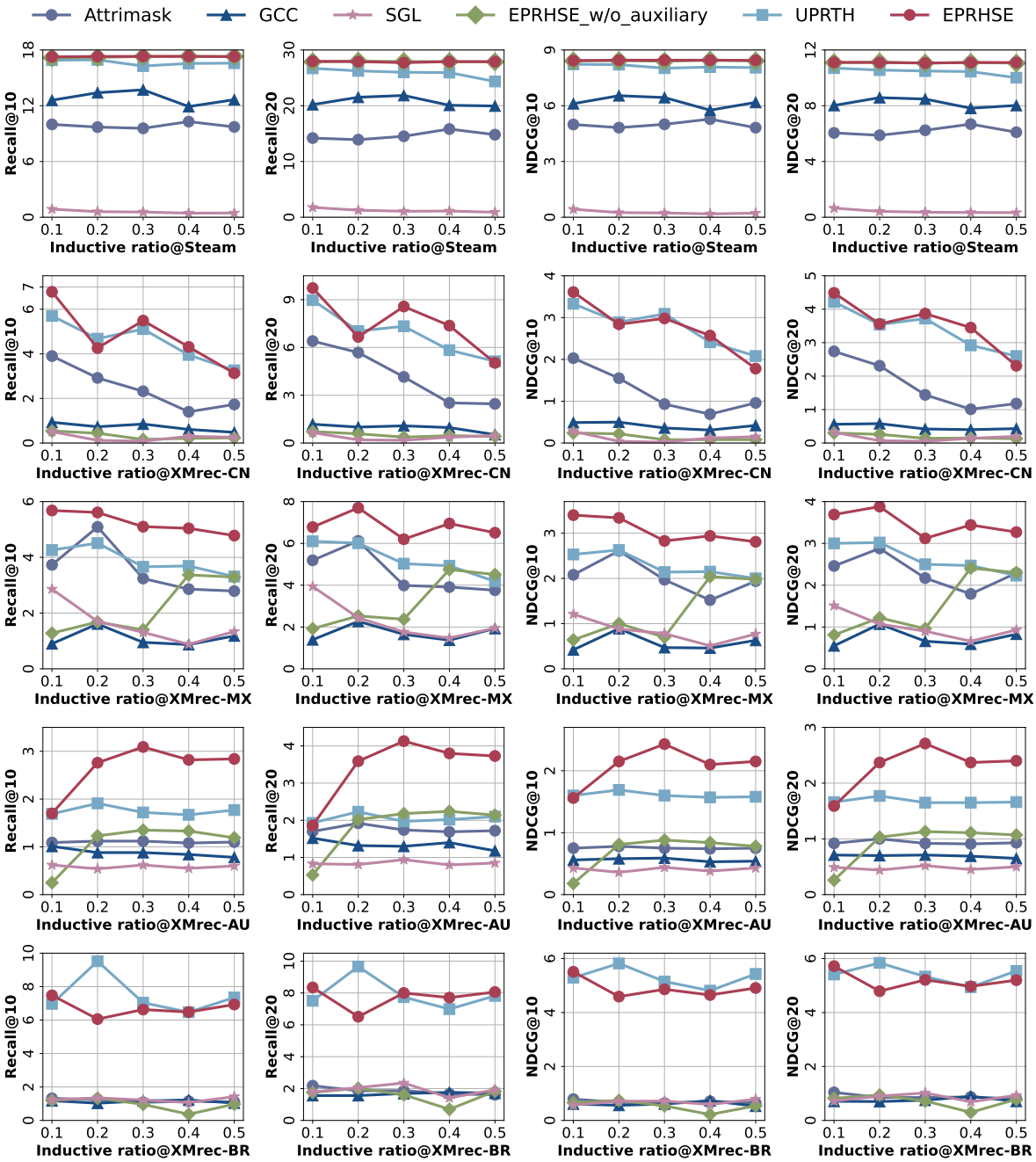


Fig. 11. Recall@{10,20} and NDCG{10,20} performance on the cold-start scenario for five datasets.

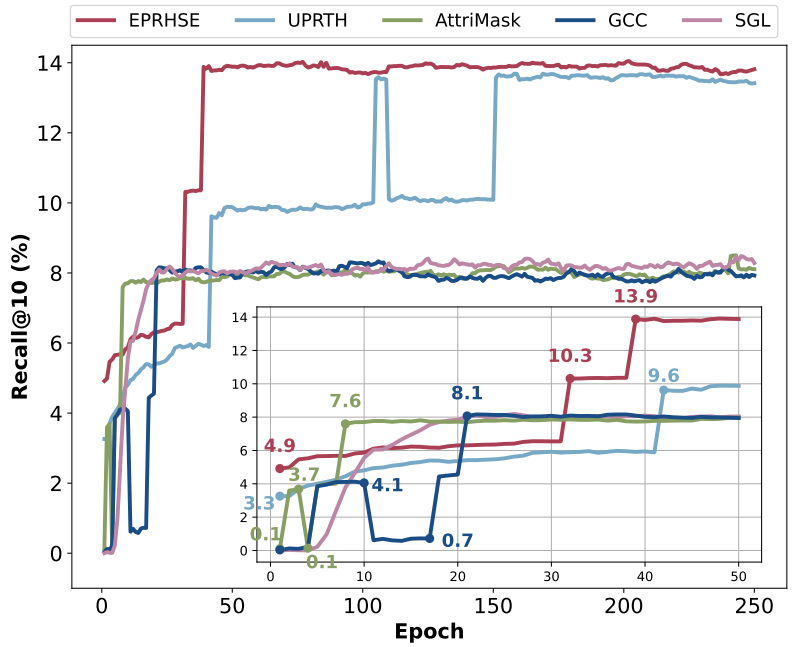
5.5 Time analysis

We assess efficiency by measuring the average runtime of EPRHSE and other models. To ensure fairness, we set the number of pre-training epochs for all pre-trained models to 800 and the number of fine-tuning epochs to 200. The total number of epochs for models without pre-training is set to 1000. For EPRHSE, we also report the runtime required to construct three-dimensional encoding trees for the user-item, item-category, and item-rating tasks. This process can be considered part of the data preparation stage, but it is not included in the model runtime mentioned earlier. As shown in Table 8, the "encoding tree" construction for the XMrec-MX and XMrec-AU datasets is executed on the CPU, as the GPU memory of our server is insufficient to handle these large datasets. Due to the addition of pooling and unpooling layers in the encoder, EPRHSE performs more extensive information aggregation per epoch compared to other HGCN-based models like HGNN, AttriMask, and UPRTH, resulting in increased runtime. Overall, the increase in runtime remains within an acceptable range. Compared to all baselines, EPRHSE exhibits a moderate runtime, being slower than LightGCN, DirectAU, UltraGCN, HGNN, HCCF, AttriMask, and UPRTH but faster than DHCF, NCL, XSimGCL, LightGCL, GCC, and SGL. Furthermore, EPRHSE outperforms all baseline models. We believe sacrificing some computational time for enhanced recommendation performance is a worthwhile trade-off.

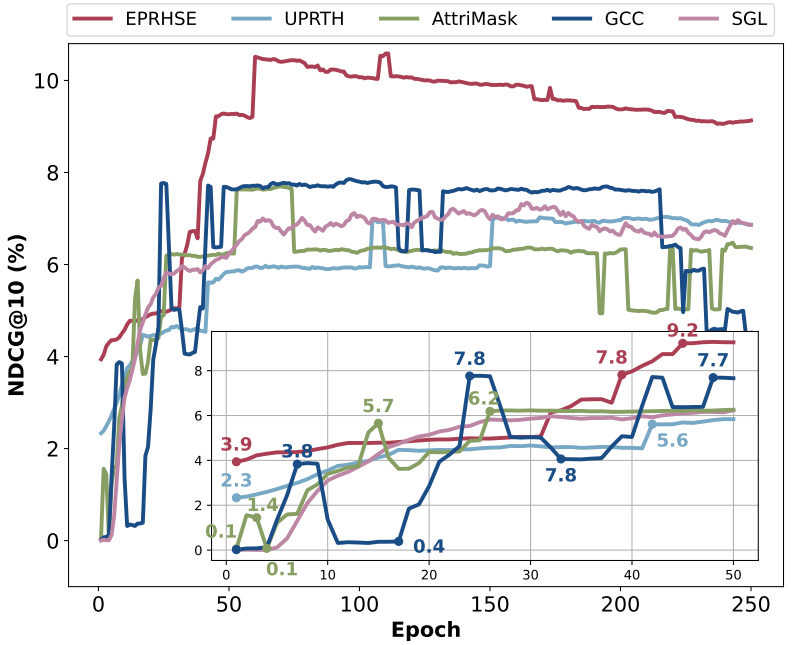
Table 8. Average time per run for EPRHSE and all baselines. The italicized results represent the time taken on the CPU, while the regular font indicates the time taken on the GPU. (Unit: Second)

Datasets	**Steam**	**XMrec-CN**	**XMrec-MX**	**XMrec-AU**	**XMrec-BR**
encoding tree	25.2	7.4	*1138.5*	*1468.4*	26.1
EPRHSE	77.2	60.1	321.9	197.1	106.3
LightGCN	37.3	15.8	72.2	128.5	51.7
DirectAU	81.1	34.6	115.5	203.6	119.6
UltraGCN	31.8	14.8	62.2	103.8	44.8
HGNN	29.7	15.4	55.0	113.7	33.0
HCCF	53.8	29.7	259.1	244.0	73.6
DHCF	273.6	120.1	568.6	777.5	292.4
NCL	468.3	205.7	2113.6	948.3	501.6
XSimGCL	172.1	75.3	1179.1	510.6	194.8
LightGCL	100.8	111.2	1876.9	692.7	132.8
GCC	255.1	207.8	300.8	290.6	212.4
SGL	508.4	433.7	434.0	516.8	435.1
AttriMask	44.1	30.7	84.3	110.8	35.0
UPRTH	68.3	43.6	248.2	174.4	94.8

The analysis of model runtime not only considers the execution time per epoch but also the convergence speed. Therefore, we compared the fine-tuning curves of five pre-trained models, including EPRHSE, to further investigate this aspect. Figure 12 illustrates the performance variation of Recall@10 and NDCG@10 over 250 fine-tuning epochs on the XMrec-BR dataset. The inset magnifies the first 50 epochs, highlighting key results for a more detailed examination. Our key findings are as follows: (1) Overall, EPRHSE exhibits a fast convergence rate while maintaining long-term stability. In contrast, UPRTH converges the slowest and experiences significant performance drops in later stages. While GCC, SGL, and AttriMask quickly reach their peak performance, their



(a) Recall@10 on XMrec-BR

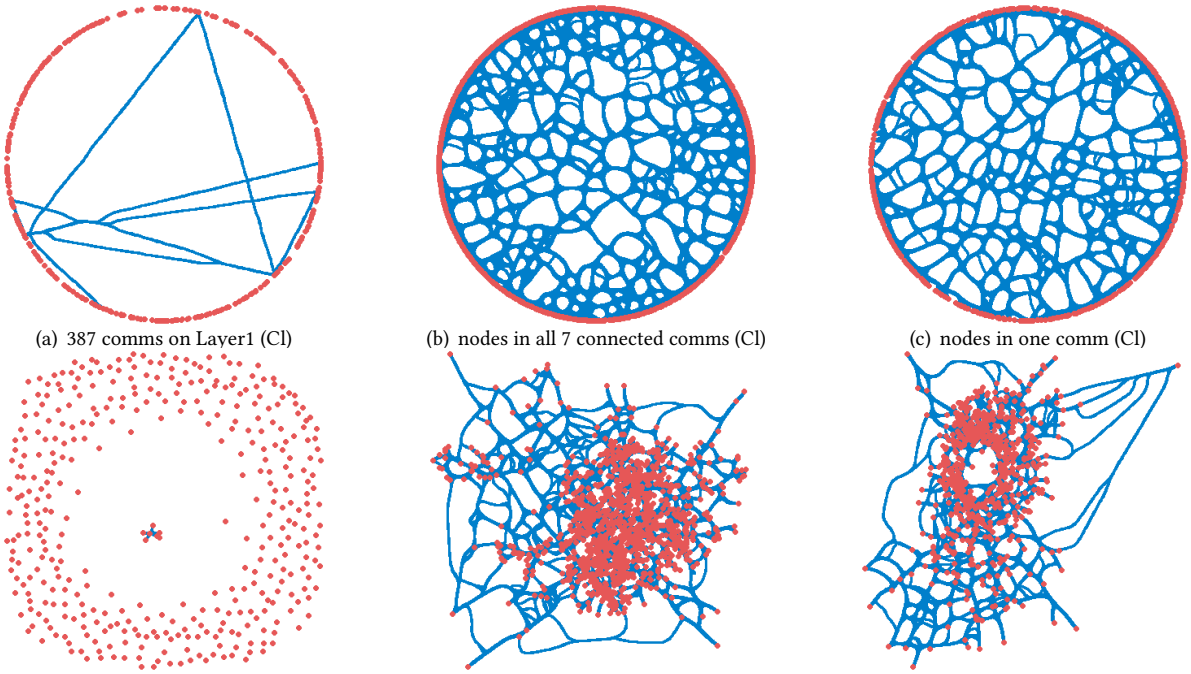


(b) NDCG@10 on XMrec-BR

Fig. 12. Recall@10 and NDCG@10 results on the XMrec-BR dataset during fine-tuning of 250 epochs.

overall effectiveness remains low. For example, NDCG@10 exhibits sharp fluctuations in the latter half of training. (2) EPRHSE, UPRTH, and SGL demonstrate a clear upward trend in the early fine-tuning stages, whereas GCC and AttriMask follow a spiral-like ascent characterized by sharp rises and falls during training. For instance, GCC's Recall@10 initially climbs to 4.1 at epoch 10 but then abruptly drops to approximately 0.7 before recovering to 8.1 around epoch 21. (3) EPRHSE starts with the highest initial performance, indicating the effectiveness of pre-training. Specifically, its Recall@10 reaches 4.9, and NDCG@10 reaches 3.9 immediately after pre-training, whereas SGL, GCC, and AttriMask only achieve around 0.1, and UPRTH achieves 3.3 and 2.3.

5.6 Visualization on Graph and Embedding

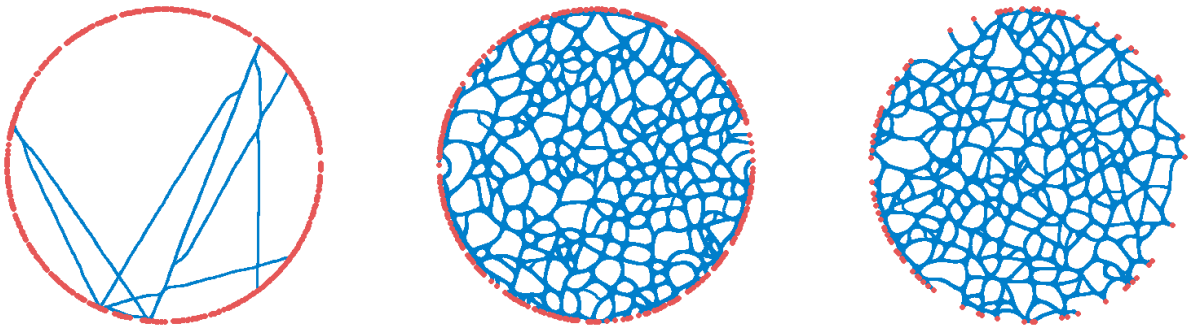


(a) 387 comms on Layer1 (Cl) (b) nodes in all 7 connected comms (Cl) (c) nodes in one comm (Cl)

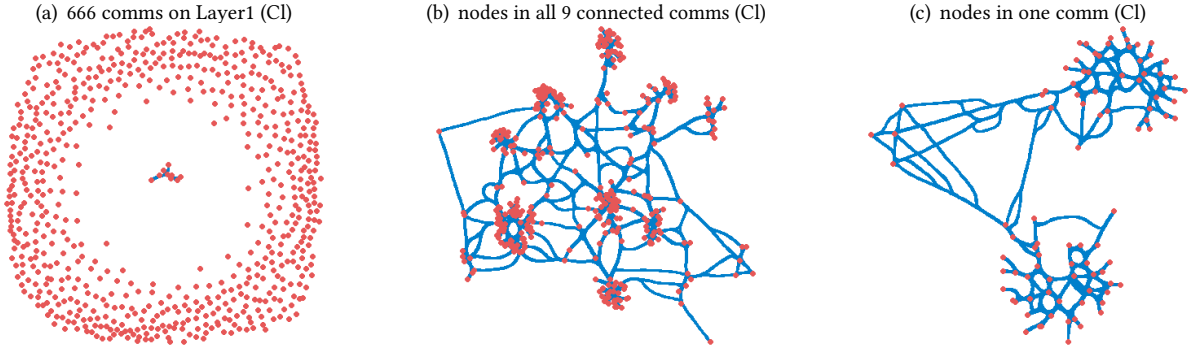
(d) 387 comms on Layer1(Fd) (e) nodes in all 7 connected comms (Fd) (f) nodes in one comm (Fd)

Fig. 13. Visualization on the Steam dataset with "item-cate" task.

5.6.1 Graph visualization. In this section, we visualize the results of hierarchical community partitioning using hypergraph structural entropy on the Steam dataset, focusing on three types of tasks: "item-cate", "item-rate", and "user-item". As shown in Figure 13, we present the visualization results for the "item-cate" task. The first row of subfigures displays the layout generated using the "Circular" layout method, while the second row shows the results using the "Force-directed" layout. Figure 13(a) illustrates the inter-community connections among the 387 communities obtained after one level of partitioning via hypergraph structural entropy. Each node represents a community, corresponding to Layer 1 in Table 6. Figure 13(b) shows the intra-community connections of all items within the seven connected communities in Figure 13(a), where each node represents an individual item. Figure 13(c) provides a zoomed-in view of a single community from Figure 13(a), depicting the internal structure among items within that community. This can be regarded as a local magnification of Figure 13(b). Figure 13(d), Figure 13(e), and Figure 13(f) present the same content as Figure 13(a), Figure 13(b), and Figure 13(c), respectively, but rendered using a 'Force-directed' layout strategy. Figure 14 presents the

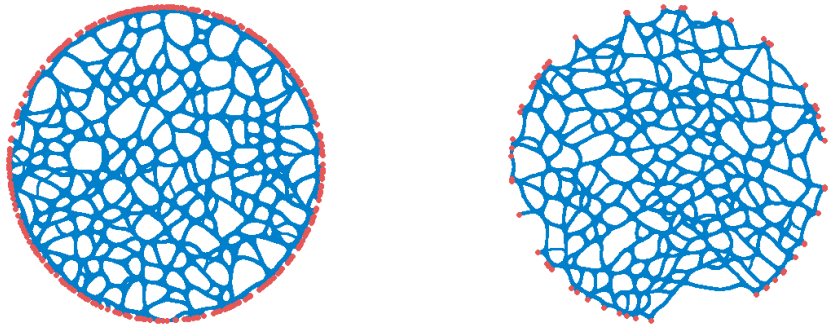


(a) 666 comms on Layer1 (Cl) (b) nodes in all 9 connected comms (Cl) (c) nodes in one comm (Cl)

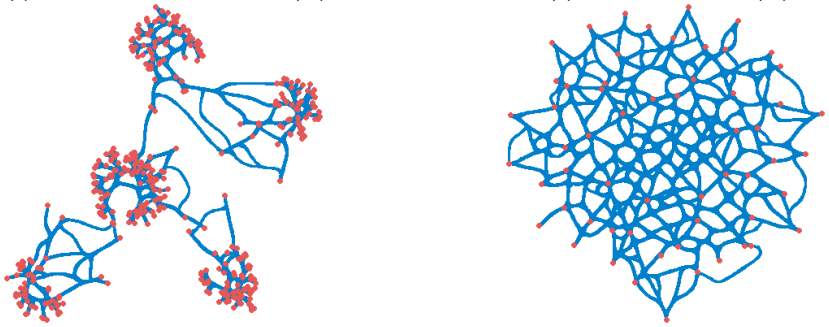


(d) 666 comms on Layer1 (Fd) (e) nodes in all 9 connected comms (Fd) (f) nodes in one comm (Fd)

Fig. 14. Visualization on the Steam dataset with "item-rate" task.



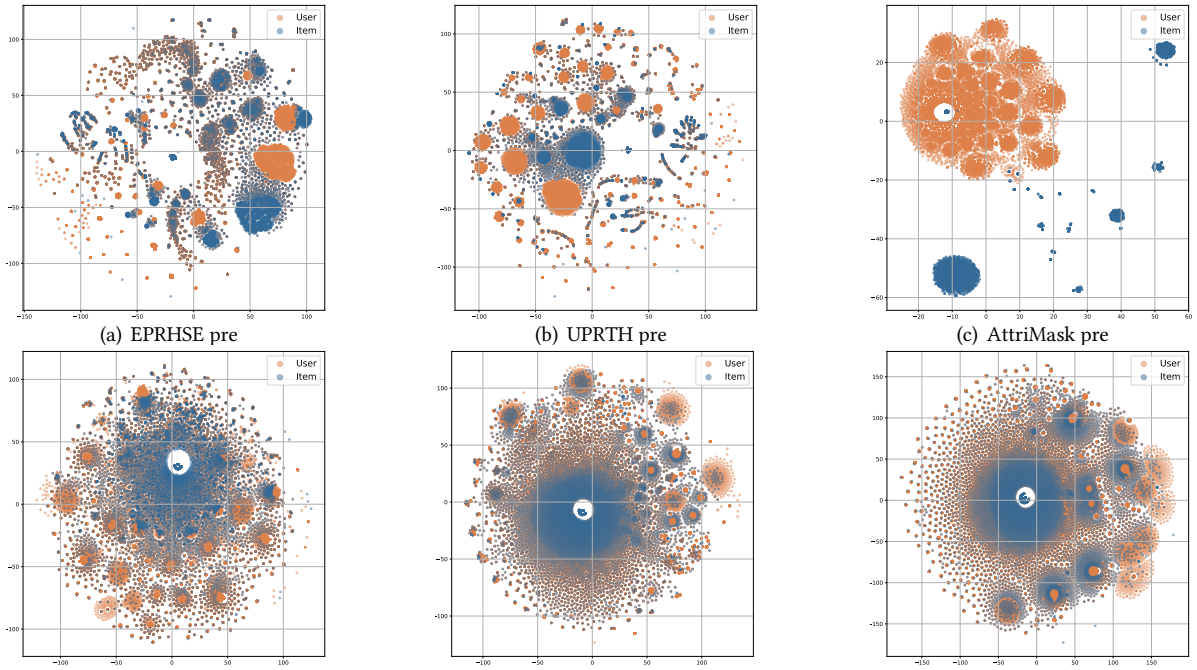
(a) nodes in 5 connected comms (Cl) (b) nodes in one comm (Cl)



(c) nodes in 5 connected comms (Fd) (d) nodes in one comm (Fd)

Fig. 15. Visualization on the Steam dataset with "user-item" task.

results for the "item-rate" task, constructed using the same methodology as Figure 13, and is therefore not discussed in detail. Figure 15 illustrates the results for the "user-item" task. As shown in Table 6, Layer 1 produces 1,726 communities, which are too numerous for direct visualization. To enhance clarity, we selected the user nodes from 5 interconnected communities for visualization in Figure 15(a) and Figure 15(c). Additionally, one of these communities is magnified to generate Figure 15(b) and Figure 15(b). The three sets of visualizations also intuitively demonstrate the effectiveness of hypergraph structural entropy in hierarchical community partitioning. It successfully groups closely connected nodes together, which facilitates the subsequent learning of group-level representations for users and items, ultimately benefiting recommendation performance.

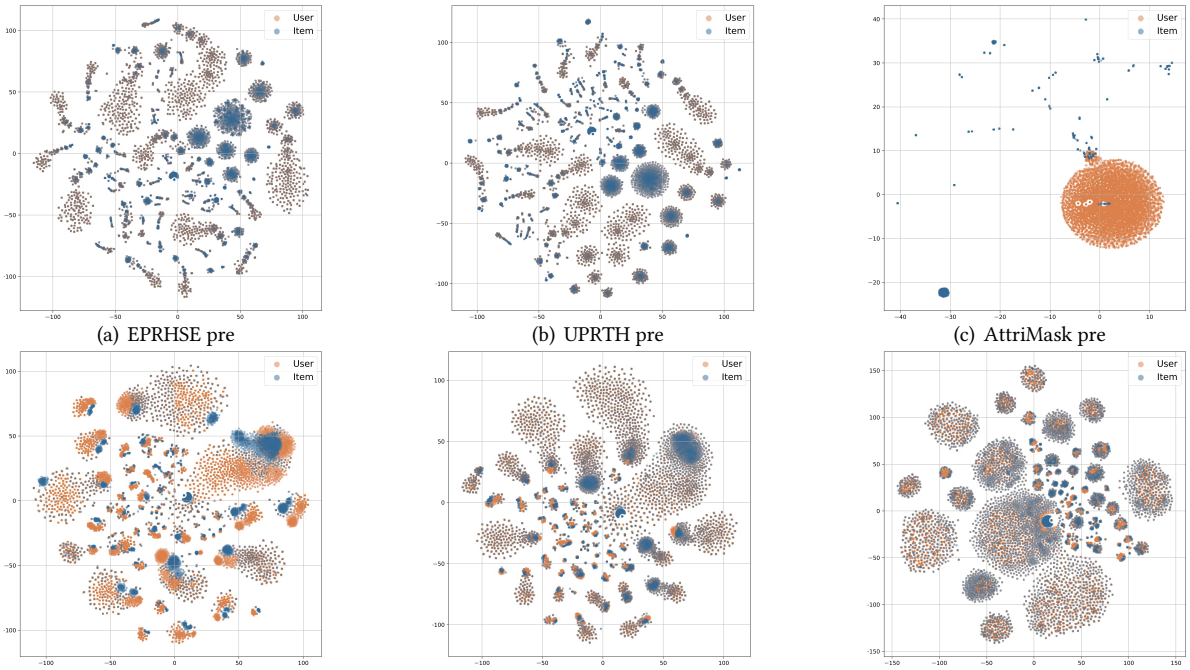


(a) EPRHSE pre (b) UPRTH pre (c) AttriMask pre

(d) EPRHSE final (e) UPRTH final (f) AttriMask final

Fig. 16. t-SNE visualization of user and item embeddings on the XMrec-BR dataset.

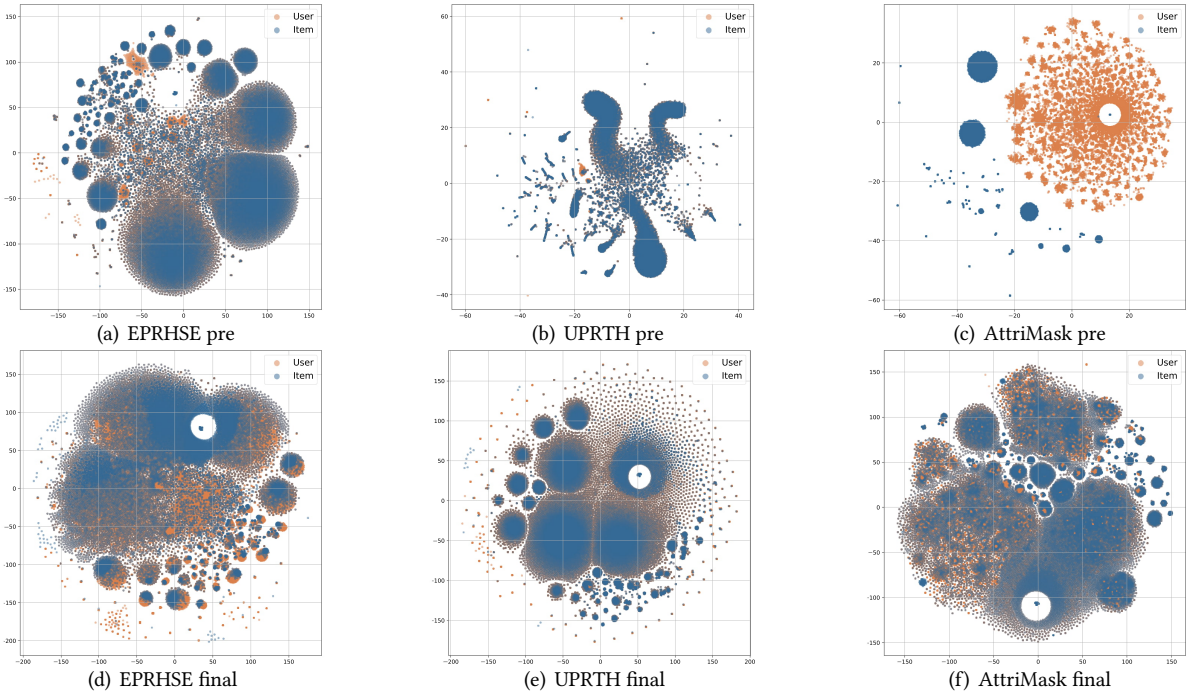
5.6.2 Embedding visualization. We compare the embedding distributions learned by EPRHSE, UPRTH, and AttriMask across five datasets, using t-SNE for two-dimensional visualization. As shown in Figure 16-Figure 20, the three small plots in the first row correspond to the embeddings obtained after pre-training, while the three plots in the second row show the final embeddings. Orange nodes represent users, and blue nodes represent items. We make the following observations: First, the embeddings obtained from EPRHSE pre-training exhibit more hierarchical community structures compared to the other two models. Specifically, they contain communities of varying sizes, relatively dispersed points, and differing proximities between communities. This is desirable, as the pre-training stage aims to integrate information from different tasks to form representative embeddings for both users and items. In contrast, UPRTH embeddings tend to be overly clustered within communities, which may result in loss of individualized information or overfitting, likely due to the TA layer fusion. AttriMask embeddings, on the other hand, show user and item embeddings that are excessively dispersed—either forming dense clusters or isolated points—resulting in extreme distributions. Second, considering the final embeddings, EPRHSE not only preserves the hierarchical community structure established during pre-training but also aligns user and



(a) EPRHSE pre (b) UPRTH pre (c) AttriMask pre

(d) EPRHSE final (e) UPRTH final (f) AttriMask final

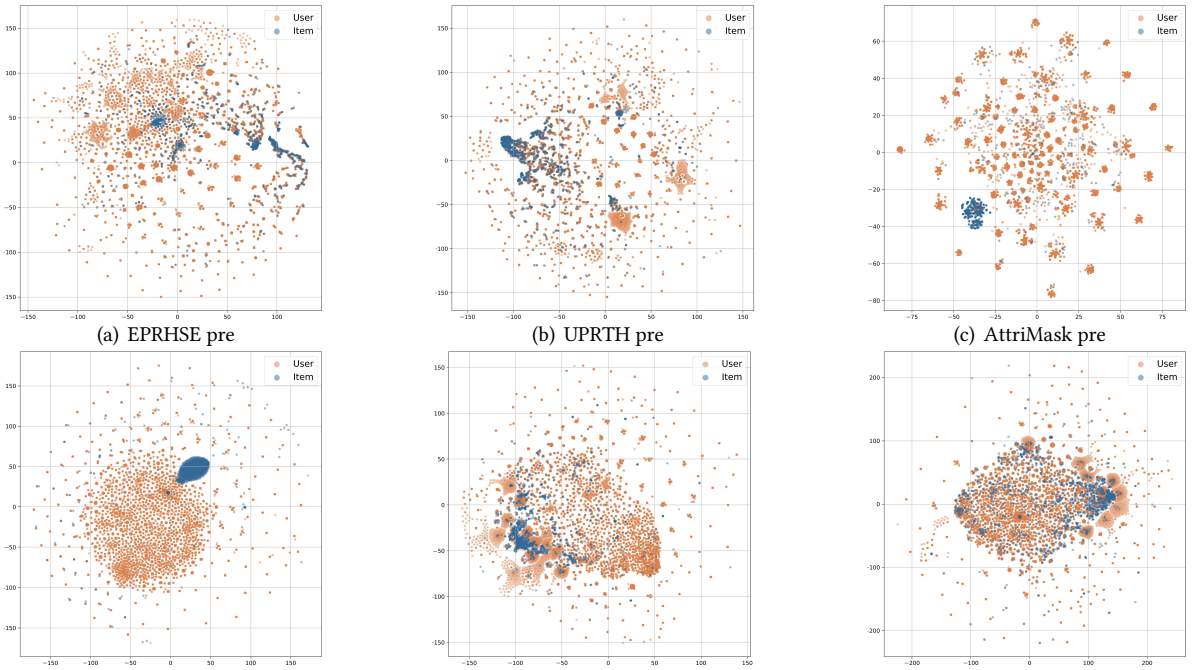
Fig. 17. t-SNE visualization of user and item embeddings on the XMrec-CN dataset.



(a) EPRHSE pre (b) UPRTH pre (c) AttriMask pre

(d) EPRHSE final (e) UPRTH final (f) AttriMask final

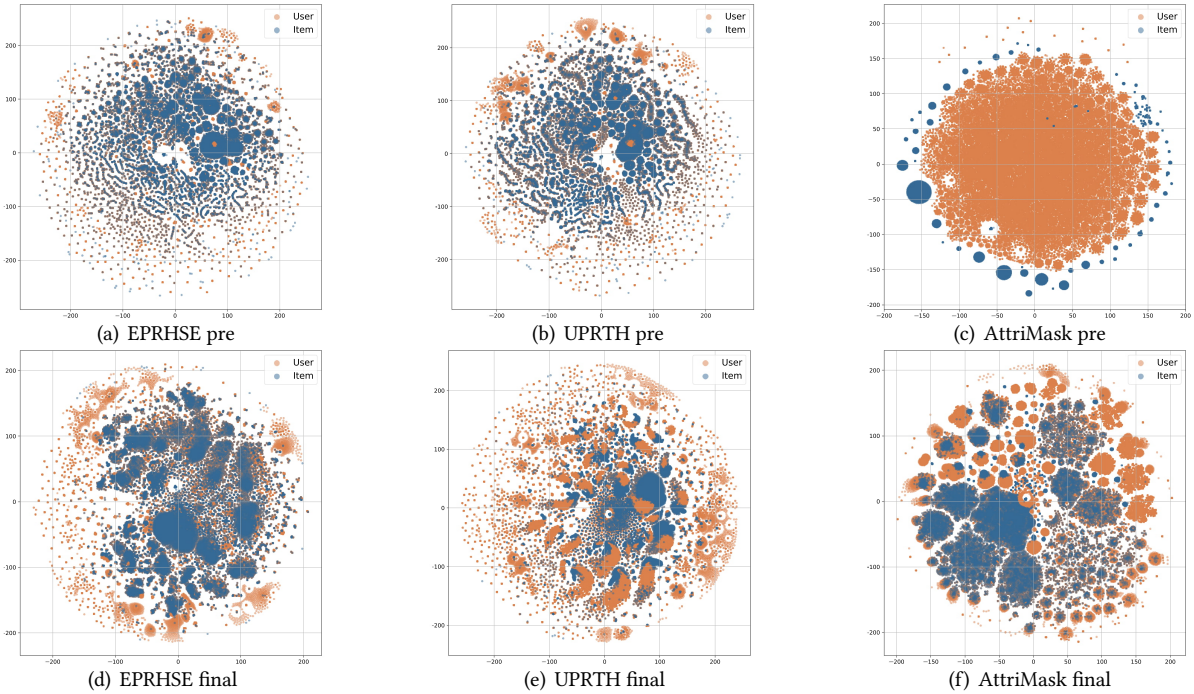
Fig. 18. t-SNE visualization of user and item embeddings on the XMrec-AU dataset.



(a) EPRHSE pre (b) UPRTH pre (c) AttriMask pre

(d) EPRHSE final (e) UPRTH final (f) AttriMask final

Fig. 19. t-SNE visualization of user and item embeddings on the Steam dataset.



(a) EPRHSE pre (b) UPRTH pre (c) AttriMask pre

(d) EPRHSE final (e) UPRTH final (f) AttriMask final

Fig. 20. t-SNE visualization of user and item embeddings on the XMrec-MX dataset.

item embeddings at the group level. This alignment facilitates top-k group-level recommendations and helps explain why EPRHSE achieves superior performance at @20. By contrast, the final embeddings of UPRTH and AttriMask either remain overly compact (forming a single dense cluster) or exhibit overly discrete, point-to-point alignment between users and items, which is detrimental for recommendation tasks.

Based on the above observations, we summarize the desirable properties of embedding distributions: good embeddings should form hierarchical communities that reflect group-level profiles without overly merging nodes so as to lose individual-specific information. Within each community, points should not be too tightly clustered, as high intra-cluster similarity can lead to redundancy. Finally, the alignment between user and item embeddings should occur at the group level rather than on a one-to-one basis, since one-to-one alignment may also introduce excessive redundancy.

6 RELATED WORK

In this section, we summarize the related literature, baselines, and approaches related to the proposed framework. It is primarily divided into three parts: recommender systems (Section 6.1), hypergraph learning (Section 6.2), and structural information theory based applications (Section 6.3).

6.1 Recommender Systems

Recurrent neural networks [33, 51, 77] are widely used for session-based recommendations due to their unique handling of time-series data. Convolutional neural networks [32, 78, 83] enhance expressive power by capturing local and global latent features. Additionally, transformers [42, 75] facilitate information interaction and transmission through attention mechanisms, modeling dependencies between users and items. Existing methods typically optimize their models by leveraging social recommendations, pre-training techniques, and knowledge graphs (KG) to address data sparsity and cold-start problems. Social recommendations involve constructing heterogeneous graphs [98, 115], hypergraphs [114], and star-structured graphs [41], etc., to predict unknown user preferences using direct or indirect social relationships and group interests [55, 79, 99, 105]. This approach helps mitigate the problem of sparse user-item interaction data [91], ultimately enhancing the quality of recommendations. Pre-training methods often supplement user representations with rich unlabeled domain content [57, 70] and fine-tune them for specific tasks to improve recommendation performance in the cold-start scenario [31, 58, 80]. Accurate recommendations require surpassing the modeling of user-item interactions and considering side information. Knowledge graphs, due to their inherent semantic relationships between entities, are often utilized as a source of side information [86, 109]. By linking items with attributes, knowledge graphs break the assumption of independent interactions [89] and facilitate fine-grained, structured relationship modeling [85, 90]. Furthermore, knowledge graphs are also used for visual reasoning to enhance the interpretability of recommendations [12, 50, 103]. New technologies like diffusion [60], LLM [54] and agent [119] nowadays are also applied for recommendation.

6.2 Hypergraph Learning

Due to its ability to connect any number of nodes with hyperedges, hypergraphs have an advantage in modeling associations and high-order relationships in data, as real-world data correlations are often more complex than pairwise relationships [27, 112]. Therefore, hypergraphs have been widely used in tasks such as classification [76], biology [28], group dynamics [19], disease prediction [73], and recommender systems [108]. Hypergraph modeling can be divided into explicit and implicit methods. Explicit modeling directly utilizes the information contained in the dataset as hyperedges, such as attribute-based and network-based methods. Implicit modeling, on the other hand, requires computational processing to obtain hyperedges, such as distance-based and representation-based

methods. This paper adopts an explicit modeling approach. Hypergraph learning aims to preserve structural and semantic information in low-dimensional vectors ideally. It can be categorized into three types: spectral learning, neighborhood preserving, and neural network-based methods. Most cases involve learning based on known hypergraph structures and node features. Dynamic hypergraph structure learning updates the hypergraph structure while optimizing the target task [121]. Additionally, research on geometric hypergraph learning [21], distributed hypergraph learning [35], hypergraph attribute learning [36], hypergraph convolution and attention [4], and pre-trained hypergraph learning [107] has enhanced representation learning capabilities. However, existing hypergraph learning faces the challenge of capturing group relationships [3]. Each hyperedge represents an indivisible relationship among a group of nodes, and such high-order relationships are difficult to capture during the embedding learning process. The proposed EPRHSE addresses this issue by learning the hierarchical structure among nodes, i.e., group high-order relationships, using hypergraph structural entropy, and utilizing this to enhance the embedding learning process.

6.3 Structural Information Theory based Applications

The Structural Information Theory decoding network's ability to capture the structure's essence has been validated in many applications. The hierarchical nature of the structure entropy encoding tree provides new methods for hierarchical structure pooling [96] and dimension estimation [111] in graph neural network, unsupervised image segmentation [104, 116], state abstraction [118] and role discovery [117] in reinforcement learning, social bot detection [67, 110], and unsupervised social event detection [11]. Additionally, reconstructing the graph structure on the hierarchical encoding tree suppresses edge noise and enhances the learning ability of the graph structure [123, 124]. Introducing structural entropy in neural networks captures the underlying connectivity graph and reduces random interference [93]. Similarly, the anchor view, guided by the principle of minimizing structural entropy, improves the performance of graph contrastive learning [95]. Furthermore, modifying the network structure based on minimizing structural entropy achieves maximum deception of community structure [56]. Based on the homogeneous graph structure entropy, the study of multi-relational graph structure entropy [10] further extends the structural information theory, making it suitable for more complex scenarios.

7 CONCLUSION

This paper investigates a new enhanced pre-training framework based on hypergraph structural entropy for recommendation. The proposed EPRHSE encodes the topology of the recommender system. By developing a pre-training framework that integrates auxiliary tasks, we have effectively captured the heterogeneous relationships between users (items) and addressed the sparsity of interactions between users and items. Furthermore, by incorporating structural information theory into hypergraph learning, we achieve hierarchical user (item) community divisions in constructing high-dimensional encoding trees, thoroughly exploring the topological structure within hypergraphs. Hypergraph pooling guides the improvement of the model architecture and allows for the extraction of global dependencies. Thus, it enhances the simulation of social diffusion processes and improves node embedding. Experimental results indicate that EPRHSE outperforms baseline models regarding Recall and NDCG while demonstrating strong stability and high performance in the cold-start scenario. Our work demonstrates the potential of structural information theory in hypergraph learning and may open new directions in hypergraph structure entropy. In the future, we aim to explore other hypergraph learning methods from a structural perspective and extend hypergraph learning to additional research domains.

ACKNOWLEDGMENTS

This work has been supported by NSFC through grants 62322202, 62441612 and 62432006, Local Science and Technology Development Fund of Hebei Province Guided by the Central Government of China through grants

246Z0102G and 254Z9902G, the Pioneer and Leading Goose R&D Program of Zhejiang through grant 2025C02044, National Key Laboratory under grant 241-HF-D07-01, Beijing Natural Science Foundation through grant L253021, Hebei Natural Science Foundation through grant F2024210008, and Major Science and Technology Special Projects of Yunnan Province through grants 202502AD080012 and 202502AD080006.

REFERENCES

[1] Muhammad Aljukhadar, Sylvain Senecal, and Charles-Etienne Daoust. 2010. Information overload and usage of recommendations. In *Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI), Barcelona, Spain*. 26–33.

[2] Muhammad Aljukhadar, Sylvain Senecal, and Charles-Etienne Daoust. 2012. Using recommendation agents to cope with information overload. *International Journal of Electronic Commerce* 17, 2 (2012), 41–70.

[3] Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. 2023. A survey on hypergraph representation learning. *Comput. Surveys* 56, 1 (2023), 1–38.

[4] Song Bai, Feihu Zhang, and Philip HS Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637.

[5] Chumki Basu, Haym Hirsh, William Cohen, et al. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the AAAI Conference*. 714–720.

[6] David Bawden and Lyn Robinson. 2009. The dark side of information: overload, anxiety and other paradoxes and pathologies. *Journal of information science* 35, 2 (2009), 180–191.

[7] Hamed Bonab, Mohammad Aliannejadi, Ali Vardasbi, Evangelos Kanoulas, and James Allan. 2021. Cross-market product recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 110–119.

[8] Desheng Cai, Shengsheng Qian, Quan Fang, Jun Hu, and Changsheng Xu. 2023. User cold-start recommendation via inductive heterogeneous graph neural network. *ACM Transactions on Information Systems* 41, 3 (2023), 1–27.

[9] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple yet effective graph contrastive learning for recommendation. In *11th International Conference on Learning Representations, ICLR 2023*.

[10] Yuwei Cao, Hao Peng, Angsheng Li, Chenyu You, Zhifeng Hao, and Philip S Yu. 2024. Multi-Relational Structural Entropy. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 1–15.

[11] Yuwei Cao, Hao Peng, Zhengtao Yu, and Philip S Yu. 2024. Hierarchical and Incremental Structural Entropy Minimization for Unsupervised Social Event Detection. In *Proceedings of the AAAI conference on artificial intelligence*. 1–13.

[12] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *Proceedings of the World Wide Web Conference*. 151–161.

[13] Jihong Chen, Wei Chen, Jinjing Huang, Jinhua Fang, Zhixu Li, An Liu, and Lei Zhao. 2020. Co-purchaser recommendation for online group buying. *Data Science and Engineering* 5 (2020), 280–292.

[14] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 335–344.

[15] Li Chen, Marco De Gemmis, Alexander Felfernig, Pasquale Lops, Francesco Ricci, and Giovanni Semeraro. 2013. Human decision making and recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 3, 3 (2013), 1–7.

[16] Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. 2024. On softmax direct preference optimization for recommendation. *Advances in Neural Information Processing Systems* 37 (2024), 27463–27489.

[17] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[18] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. 271–280.

[19] Guilherme Ferraz de Arruda, Giovanni Petri, and Yamir Moreno. 2020. Social contagion models on hypergraphs. *Physical Review Research* 2, 2 (2020), 023032.

[20] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[21] Dawei Du, Honggang Qi, Longyin Wen, Qi Tian, Qingming Huang, and Siwei Lyu. 2016. Geometric hypergraph learning for visual tracking. *IEEE Transactions on Cybernetics* 47, 12 (2016), 4182–4195.

[22] Mark H Ebell, Jay Siwek, Barry D Weiss, Steven H Woolf, Jeffrey Susman, Bernard Ewigman, and Marjorie Bowman. 2004. Strength of recommendation taxonomy (SORT): a patient-centered approach to grading evidence in the medical literature. *The Journal of the American Board of Family Practice* 17, 1 (2004), 59–67.

[23] Angela Edmunds and Anne Morris. 2000. The problem of information overload in business organisations: a review of the literature. *International journal of information management* 20, 1 (2000), 17–28.
[24] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference*. 417–426.
[25] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference*, Vol. 33. 3558–3565.
[26] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering* 19, 3 (2007), 355–369.
[27] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. 2020. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 5 (2020), 2548–2566.
[28] Sathyanarayanan Gopalakrishnan, Supriya Sridharan, Soumya Ranjan Nayak, Janmenjoy Nayak, and Swaminathan Venkataraman. 2022. Central hubs prediction for bio networks by directed hypergraph-GA with validation to COVID-19 PPI. *Pattern Recognition Letters* 153 (2022), 246–253.
[29] Zhiqiang Guo, Jianjun Li, Guohui Li, Chaoyang Wang, Si Shi, and Bin Ruan. 2024. Lgmrec: Local and global graph learning for multimodal recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8454–8462.
[30] Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, and Erel Uziel. 2010. Social media recommendation based on people and tags. In *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval*. 194–201.
[31] Bowen Hao, Hongzhi Yin, Jing Zhang, Cuiping Li, and Hong Chen. 2023. A multi-strategy-based pre-training method for cold-start recommendation. *ACM Transactions on Information Systems* 41, 2 (2023), 1–24.
[32] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.
[33] Balazs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. (2016), 1–10.
[34] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *Proceedings of the ICLR*. 1–22.
[35] Jin Huang, Rui Zhang, and Jeffrey Xu Yu. 2015. Scalable hypergraph learning and processing. In *Proceedings of the 2015 IEEE International Conference on Data Mining*. IEEE, 775–780.
[36] Sheng Huang, Mohamed Elhoseiny, Ahmed Elgammal, and Dan Yang. 2015. Learning hypergraph-regularized attribute predictors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 409–417.
[37] Thomas W Jackson and Pourya Farzaneh. 2012. Theory-based model of factors affecting information overload. *International Journal of Information Management* 32, 6 (2012), 523–532.
[38] Jacob Jacoby. 1984. Perspectives on information overload. *Journal of consumer research* 10, 4 (1984), 432–435.
[39] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*. 135–142.
[40] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In *Proceedings of the ACM SIGKDD Conference on Machine Learning*. 20201–2029.
[41] Meng Jiang, Peng Cui, Fei Wang, Qiang Yang, Wenwu Zhu, and Shiqiang Yang. 2012. Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. 1422–1431.
[42] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE 2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.
[43] Bilal Khan, Jia Wu, Jian Yang, and Xiaoxiao Ma. 2023. Heterogeneous hypergraph neural network for social recommendation using attention network. *ACM Transactions on Recommender Systems* (2023).
[44] Shristi Shakya Khanal, PWC Prasad, Abeer Alsadoon, and Angelika Maag. 2020. A systematic review: machine learning based recommendation systems for e-learning. *Education and Information Technologies* 25, 4 (2020), 2635–2664.
[45] Sherrie YX Komiak and Izak Benbasat. 2006. The effects of personalization and familiarity on trust and adoption of recommendation agents. *MIS quarterly* (2006), 941–960.
[46] Zhiyu Kong, Xiaoru Zhang, and Ruilin Wang. 2021. Review of the Research on the Relationship Between Algorithmic News Recommendation and Information Cocoons. In *Proceedings of the 3rd International Conference on Literature, Art and Human Development (ICLAHD 2021)*. Atlantis Press, 341–345.
[47] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*. 1748–1757.
[48] Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory* 62, 6 (2016), 3290–3339.

[49] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 2615–2623.
[50] Haotian Li, Yong Wang, Songheng Zhang, Yangqiu Song, and Huamin Qu. 2021. KG4Vis: A knowledge graph-based approach for visualization recommendation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 195–205.
[51] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. 1419–1428.
[52] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World Wide Web*. 661–670.
[53] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM web conference 2022*. 2320–2329.
[54] Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. 2024. Mamba4rec: Towards efficient sequential recommendation with selective state space models. *arXiv preprint arXiv:2403.03900* (2024).
[55] Huafeng Liu, Liping Jing, Jian Yu, and Michael K Ng. 2019. Social recommendation with learning personal and social latent factors. *IEEE Transactions on Knowledge and Data Engineering* 33, 7 (2019), 2956–2970.
[56] Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. REM: From structural entropy to community structure deception. *Proceedings of the Advances in Neural Information Processing Systems* 32 (2019), 1–11.
[57] Yong Liu, Susen Yang, Chenyi Lei, Guoxin Wang, Haihong Tang, Juyong Zhang, Aixin Sun, and Chunyan Miao. 2021. Pre-training graph transformer with multimodal side information for recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 2853–2861.
[58] Zhiwei Liu, Ziwei Fan, Yu Wang, and Philip S Yu. 2021. Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer. In *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval*. 1608–1612.
[59] Nicholas H Lurie. 2004. Decision making in information-rich environments: The role of information structure. *Journal of consumer research* 30, 4 (2004), 473–486.
[60] Haokai Ma, Ruobing Xie, Lei Meng, Xin Chen, Xu Zhang, Leyu Lin, and Zhanhui Kang. 2024. Plug-in diffusion model for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 8886–8894.
[61] Lyu Michael R Ma Hao, Yang Haixuan and King Irwin. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 931–940.
[62] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the ACM CIKM*. 1253–1262.
[63] Mark O'Neill, Elham Vaziripour, Justin Wu, and Daniel Zappala. 2016. Condensing steam: Distilling the diversity of gamer behavior. In *Proceedings of the 2016 Internet Measurement Conference*. 81–95.
[64] Charles Oppenheim. 1997. Managers' use and handling of information. *International journal of information management* 17, 4 (1997), 239–248.
[65] Denis Parra, Alexandros Karatzoglou, Xavier Amatriain, Idil Yavuz, et al. 2011. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. *Proceedings of the CARS-2011* 5 (2011).
[66] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 325–341.
[67] Hao Peng, Jingyun Zhang, Xiang Huang, Zhifeng Hao, Angsheng Li, Zhengtao Yu, and Philip S Yu. 2024. Unsupervised Social Bot Detection via Structural Information Theory. *ACM Transactions on Information Systems* (2024).
[68] Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. 2011. Terms of a Feather: Content-based news recommendation and discovery using Twitter. In *Proceedings of the European Conference on Information Retrieval*. Springer, 448–459.
[69] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the ACM SIGKDD*. 1150–1160.
[70] Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-BERT: Pre-training user representations for improved recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4320–4327.
[71] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World Wide Web*. 811–820.
[72] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.
[73] Wei Shao, Yao Peng, Chen Zu, Mingliang Wang, Daoqiang Zhang, Alzheimer's Disease Neuroimaging Initiative, et al. 2020. Hypergraph based multi-task feature selection for multimodal classification of Alzheimer's disease. *Computerized Medical Imaging and Graphics* 80 (2020), 101663.

[74] Andrew J Stanley and Philip S Clipsham. 1997. Information overload-myth or reality?. In *IEE Colloquium on IT Strategies for Information Overload (Digest No: 1997/340)*. IET, 1–1.

[75] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[76] Liang Sun, Shuiwang Ji, and Jieping Ye. 2008. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 668–676.

[77] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the DLRS1st workshop on deep learning for recommender systems*. 17–22.

[78] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the ACM international conference on web search and data mining*. 565–573.

[79] Jiliang Tang, Suhang Wang, Xia Hu, Dawei Yin, Yingzhou Bi, Yi Chang, and Huan Liu. 2016. Recommendation with social dimensions. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[80] Changxin Tian, Zihan Lin, Shuqing Bian, Jinpeng Wang, and Wayne Xin Zhao. 2022. Temporal contrastive pre-training for sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1925–1934.

[81] Haoye Tian, Haini Cai, Li Shun Wen, Junhao, and Yingqiao Li. 2019. A music recommendation system based on logistic regression and eXtreme gradient boosting. In *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.

[82] Jason Turcotte, Chance York, Jacob Irving, Rosanne M Scholl, and Raymond J Pingree. 2015. News recommendations from social media opinion leaders: Effects on media trust and information seeking. *Journal of computer-mediated communication* 20, 5 (2015), 520–535.

[83] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. *Advances in neural information processing systems* 26 (2013).

[84] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representnternation alignment and uniformity in collaborative filtering. In *Proceedings of the ACM SIGKDD*. 1816–1825.

[85] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 1835–1844.

[86] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *Proceedings of the World Wide Web Conference*. 2000–2010.

[87] Jinpeng Wang, Ziyun Zeng, Yunxiao Wang, Yuting Wang, Xingyu Lu, Tianxiang Li, Jun Yuan, Rui Zhang, Hai-Tao Zheng, and Shu-Tao Xia. 2023. Missrec: Pre-training and transferring multi-modal interest-aware sequence representation for recommendation. In *Proceedings of the 31st ACM International Conference on Multimedia*. 6548–6557.

[88] Nan Wang, Dan Liu, Jin Zeng, Lijin Mu, and Jinbao Li. 2024. HGRec: Group recommendation with hypergraph convolutional networks. *IEEE Transactions on Computational Social Systems* (2024).

[89] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.

[90] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web conference 2021*. 878–887.

[91] Xin Wang, Wei Lu, Martin Ester, Can Wang, and Chun Chen. 2016. Social recommendation with strong and weak ties. In *Proceedings of the 25th ACM International On conference on Information and Knowledge Management*. 5–14.

[92] Yaozheng Wang, Dawei Feng, Dongsheng Li, Xinyuan Chen, Yunxiang Zhao, and Xin Niu. 2016. A mobile recommendation system based on logistic regression and gradient boosting decision trees. In *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1896–1902.

[93] Yifei Wang, Yupan Wang, Zeyu Zhang, Song Yang, Kaiqi Zhao, and Jiamou Liu. 2023. User: Unsupervised structural entropy-based robust graph neural network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 10235–10243.

[94] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*. 3597–3606.

[95] Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, and Ke Xu. 2023. SEGA: Structural Entropy Guided Anchor View for Graph Contrastive Learning. In *Proceedings of the ICML*. PMLR, 1–20.

[96] Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. 2022. Structural entropy guided graph hierarchical pooling. In *Proceedings of the International Conference on Machine Learning*. PMLR, 24017–24030.

[97] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the ACM SIGIR*. 726–735.

[98] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4753–4766.

[99] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 235–244.

[100] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[101] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[102] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *Proceedings of the ACM SIGIR*. 70–79.

[103] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 285–294.

[104] Minhui Xie, Hao Peng, Pu Li, Guangjie Zeng, Shuhai Wang, Jia Wu, Peng Li, and Philip S Yu. 2025. Hierarchical Superpixel Segmentation via Structural Information Theory. *Proceedings of The SIAM International Conference on Data Mining*.

[105] Fei Xiong, Ximeng Wang, Shirui Pan, Hong Yang, Haishuai Wang, and Chengqi Zhang. 2018. Social recommendation with evolutionary opinion dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 10 (2018), 3804–3816.

[106] Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. 2023. Spatio-temporal hypergraph learning for next POI recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*. 403–412.

[107] Mingdai Yang, Zhiwei Liu, Liangwei Yang, Xiaolong Liu, Chen Wang, Hao Peng, and Philip S Yu. 2024. Instruction-based Hypergraph Pretraining. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[108] Mingdai Yang, Zhiwei Liu, Liangwei Yang, Xiaolong Liu, Chen Wang, Hao Peng, and Philip S Yu. 2024. Unified Pretraining for Recommendation via Task Hypergraphs. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 891–900.

[109] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *Proceedings of the international ACM SIGIR conference on research and development in information retrieval*. 1434–1443.

[110] Yingguang Yang, Qi Wu, Buyun He, Hao Peng, Renyu Yang, Zhifeng Hao, and Yong Liao. 2024. SeBot: Structural Entropy Guided Multi-View Contrastive Learning for Social Bot Detection. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[111] Zhenyu Yang, Ge Zhang, Jia Wu, Jian Yang, Quan Z Sheng, Hao Peng, Angsheng Li, Shan Xue, and Jianlin Su. 2023. Minimum entropy principle guided graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 114–122.

[112] Jun Yu, Dacheng Tao, and Meng Wang. 2012. Adaptive hypergraph learning and its application in image classification. *IEEE Transactions on Image Processing* 21, 7 (2012), 3262–3272.

[113] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 36, 2 (2023), 913–926.

[114] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the Web conference 2021*. 413–424.

[115] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 283–292.

[116] Guangjie Zeng, Hao Peng, Angsheng Li, Zhiwei Liu, Chunyang Liu, S Yu Philip, and Lifang He. 2023. Unsupervised Skin Lesion Segmentation via Structural Entropy Minimization on Multi-Scale Superpixel Graphs. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 768–777.

[117] Xianghua Zeng, Hao Peng, and Angsheng Li. 2023. Effective and stable role-based multi-agent collaboration by structural information principles. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11772–11780.

[118] Xianghua Zeng, Hao Peng, Angsheng Li, Chunyang Liu, Lifang He, and Philip S. Yu. 2023. Hierarchical State Abstraction based on Structural Information Principles. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4549–4557.

[119] An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024. On generative agents in recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in Information Retrieval*. 1807–1817.

[120] Yipeng Zhang, Xin Wang, Hong Chen, and Wenwu Zhu. 2023. Adaptive disentangled transformer for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 3434–3445.

[121] Zizhao Zhang, Haojie Lin, Yue Gao, and KLISS BNRist. 2018. Dynamic hypergraph structure learning.. In *Proceedings of the IJCAI*. 3162–3169.

[122] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.

[123] Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. 2023. SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization. In *Proceedings of the ACM Web Conference 2023*. 499–510.

[124] Dongcheng Zou, Senzhang Wang, Xuefeng Li, Hao Peng, Yuandong Wang, Chunyang Liu, Kehua Sheng, and Bo Zhang. 2024. Multispans: A multi-range spatial-temporal transformer network for traffic forecast via structural entropy optimization. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 1–10.