©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Structure- and Logic-aware Heterogeneous Graph Learning for Recommendation

Anchen Li[1,2,5], Bo Yang[1,2,*], Huan Huo[3], Farookh Khadeer Hussain[3], Guandong Xu[4]
College of Computer Science and Technology, Jilin University, China[1]
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China[2]
School of Computer Science, University of Technology Sydney, Australia[3]
Education University of Hong Kong, Hong Kong, China[4]; Aalto University, Finland[5]
liac@jlu.edu.cn, ybo@jlu.edu.cn, huan.huo@uts.edu.au, farookh.hussain@uts.edu.au, gdxu@eduhk.hk

***Abstract*—Recently, there has been a surge in recommendations based on heterogeneous information networks (HINs), attributed to their ability to integrate complex and rich semantics. Despite this advancement, most HIN-based recommenders overlook two critical aspects. First, they often fail to consider HIN's heterophily nature, hindering the capture of non-local structures in HINs. Second, most methods lack the capability for logical reasoning. In this paper, we propose a novel structure- and logic-aware heterogeneous graph learning framework for recommender systems (SLHRec). Our SLHRec contains a structure-aware module and a logic-aware module. The former uses network geometry to construct non-local neighborhoods for nodes in HINs, and then introduces a graph neural network to integrate constructed neighbors for modeling the heterophily of HINs. The logic-aware module uses the Markov logic network (MLN) to infuse logic rules into heterogeneous graph learning, thereby boosting logic reasoning in recommendations. Furthermore, we utilize contrastive learning to model cooperative signals between modules, enabling them to complement each other. In the prediction stage, both modules contribute to generating recommendations. Compared with several strong recommender baselines, our SLHRec achieves superior performance on four real-world datasets.**

***Index Terms*—Recommendation, Heterogeneous Graph, Graph Neural Networks, Markov Logic Network, Contrastive Learning.**

I. Introduction

In the age of information overload, recommender systems have become indispensable tools for various online platforms, e.g., Amazon, Bing, and Netflix. Traditional recommendation approaches, such as matrix factorization [26], primarily focus on modeling user preferences derived from their past interactions with items. However, with the growing availability of auxiliary data, there's been a notable enhancement in recommendation performance. The emergence of heterogeneous information networks (HINs), which contain multiple types of nodes and links, has proven to be an effective way to integrate complex data [53]. This data modeling has enhanced many recommenders, known as HIN-based recommendations [35].

Recent applications of the HIN in recommender systems can be categorized into two primary methods. The first school uses metapaths to establish a connection between users and items, aiding in the learning of user and item embeddings [21], [52], [62]. The second school involves graph-based methodologies [9], [29], [58], [60], employing graph neural networks (GNNs) to learn user and item embeddings by aggregating neighborhood information in HINs. To generate recommendations, both methods often use a matching function, such as inner product or neural networks, to process refined user and item embeddings. Despite the proven efficacy of these methods, they commonly overlook two crucial aspects.

Firstly, most methods fail to consider the HINs' heterophily nature, which limits their ability to capture non-local structures in HINs. In the graph field, heterophily is a distinct network concept from the heterogeneous property. Heterophily refers to nodes with similar features or the same classes are distant from each other in the graph. Our empirical study (cf. Subsection IV-B) shows this heterophily in HINs. Previous graph learning research [46], [80] shows the value of recognizing and using meaningful non-local dependencies between nodes to model the heterophily of graphs. However, most recommenders focus on node original neighbors in HINs (i.e., local structures), neglecting the insights from distant but informative nodes (i.e., non-local structures). Metapath-based approaches, as pointed out by [52], depend on path reachability, potentially missing non-local but meaningful structures. Moreover, these methods underperform when metapaths are sparse or noisy [23]. For graph-based methods, while employing multiple GNN layers appears as a solution, it introduces the risk of overshadowing vital information with redundant messages from proximal nodes [46]. In addition, as mentioned in [39], excessive layer stacking may also lead to issues such as over-smoothing [7] and over-squashing [2]. Thus, to consider the inherent structural properties of HINs in recommendation, it is important to go beyond traditional methods that focus only on node local neighbors. Considering non-local dependencies is also crucial.

Secondly, most HIN-based recommenders [9], [21], [52] do not adequately consider the logical relationships during heterogeneous graph learning. They largely hinge on a matching principle. Specifically, these methods usually match entity (e.g., users and items) embeddings through either predefined or learned similarity functions to capture and represent associative relevance patterns in HINs. However, the recommendation process is more cognitive than perceptual [8], [55]. It requires not only pattern learning and matching but also cognitive reasoning. Consider a book recommendation scenario: if a

*Corresponding Author: Bo Yang.

user expresses a preference for a certain author, it is logical to assume she would enjoy books written by that author. This reasoning follows the logical rule: *LikeAuthor*$(u, a) \wedge$ *WriteBook*$(a, b) \Rightarrow$ *LikeBook*(u, b). The community has acknowledged the need to shift AI from perception-based tasks to those requiring cognition [4], [8], [40], [55], [75], [76]. Therefore, an ideal HIN-based recommender approach should extend beyond mere recognition of relevance patterns to encompass the logical reasoning capability, thereby ensuring effective personalized recommendations.

Motivated by these gaps, we propose a comprehensive HIN-based recommender method, which uses the HIN structural information and incorporates HIN-relevant logic rules. However, such a task faces several challenges: (i) Non-local structure dependencies, essential for modeling heterophily, are latent in the HIN. The first challenge is devising an algorithm to extract these dependencies effectively. (ii) As mentioned in [46], local and non-local structures are complementary to graph learning. The second challenge lies in developing an elegant approach to model this structural collaborative relation in HINs. (iii) Logic rules could sometimes be imperfect or even contradictory. For instance, while a preference for a particular author might generally indicate an affinity for their books, exceptions to this rule could happen. Therefore, how to model the uncertainty in logic rules is the third challenge. (iv) Using HIN's structural information and related logic rules represent two distinct modeling methods. The fourth challenge lies in harmonizing these modeling perspectives to ensure they work cooperatively and enhance the recommendation.

In light of these challenges, we propose a structure- and logic-aware heterogeneous graph learning framework for recommendation (SLHRec). Our SLHRec consists of a structure-aware module and a logic-aware module. The structure-aware module learns the HIN at a structural level. To uncover non-local dependencies, it uses the idea of the neighbor extension [80]. Specifically, the module first transforms HINs into a latent space via network geometry to construct non-local dependency neighbors for nodes. Then, to model node original and constructed neighbors, we design a dual graph neural network equipped with delicate information transfer units. These units facilitate the knowledge sharing between original and constructed neighbor modelings. The logic-aware module utilizes the Markov logic network (MLN) [49] to model the logical relations related to the HIN. Specifically, it defines the joint distribution of possible facts in HINs by MLNs with logic rules and employs a heterogeneous graph inference network to predict the posterior distribution of the unobserved facts in HINs. MLNs learn the weights of logic rules in a probabilistic framework and thus could soundly handle the uncertainty in logic rules. Additionally, we introduce a self-supervised task to model the cooperative association between modules. This task treats two modules as different contrastive views, using contrastive learning to boost the representation learning ability of these viewpoints by fostering cross-view alignment. Consequently, representations from different views can learn from each other and achieve mutual enhancement.

When generating recommendations, our method integrates insights from both modules to output enhanced results. We evaluate the effectiveness of our SLHRec utilizing four datasets. The results reveal that SLHRec outperforms strong baselines in click-through rate prediction and top-K recommendation. Our contributions are as follows:

- We propose a HIN-based recommender SLHRec to consider the heterophily nature and logical relations in HINs.
- We develop a novel dual graph neural network to capture local and non-local structures in HINs for recommendation.
- We merge the Markov logic network with the heterogeneous graph embedding to model HINs at a logical level.
- We devise a self-supervised task to enhance the cooperative association between the structure and logic views of HINs.

II. Preliminaries

This section introduces some definitions and formulates the problem. Main notations are summarized in Table I.

TABLE I
SUMMARY OF KEY NOTATIONS.

Symbol	Explanation
$\mathcal{G}$	A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$\mathcal{V}, \mathcal{E}$	Node set and edge set in a graph
$\mathcal{V}_U, \mathcal{V}_I$	User entity set and item entity set in a graph
$\mathbf{Y}$	User-item interaction matrix
$\mathbf{e}_v$	Initial embedding of node $v \in \mathcal{V}$
$\mathbf{e}_v^{\mathrm{ST}}$	Output embedding from structure-aware module of $v \in \mathcal{V}$
$\mathbf{e}_v^{\mathrm{LO}}$	Output embedding from logic-aware module of $v \in \mathcal{V}$
$\mathcal{N}_v^o$	Original neighborhood of node $v \in \mathcal{V}$ in the HIN
$\mathcal{N}_v^s$	Constructed non-local neighborhood of node $v \in \mathcal{V}$
$\mathcal{R}_o$	Original relation set in the HIN
$\mathcal{R}_s$	Geometric relation set for non-local neighborhood
$\mathcal{F}$	Logic rule set of the HIN
$\mathcal{T}$	Fact set of the HIN
$\mathcal{T}_\mathcal{O}, \mathcal{T}_\mathcal{H}$	Observed fact set and hidden fact set of the HIN
$M(t)$	Markov blanket of fact $t \in \mathcal{T}$
$\hat{y}_{ui}$	Predicted value of item i by user u

A. Graph Neural Network

Graph neural networks (GNNs) offer an efficient approach to model the graph data. In our SLHRec, this technique is instrumental within the structure-aware module. A hallmark of GNNs is their neighborhood aggregation strategy. This iterative process refines the representation of a node by aggregating the representations of its neighboring nodes. Formally, in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$, the node neighborhood aggregation of a GNN is described as:

$$\mathbf{v}_a^{(l)} = \mathrm{Aggre}(\mathbf{v}_b^{(l-1)} | v_b \in \mathcal{N}_a), \tag{1}$$

where $\mathbf{v}_a^{(l)}$ denotes the representation of node $v_a \in \mathcal{V}$ at the l-th layer, with $\mathbf{v}_a^{(0)}$ representing v_a's initial feature. The set $\mathcal{N}_a$ refers to the neighboring nodes of v_a. The choice of the Aggre function is important in designing GNN, leading to the proposal of various GNN architectures [18], [25], [57].

After stacking L layers of the GNN, a Readout function may be used to output the final node representation:

$$\mathbf{v}_a^{(l)} = \text{Readout}(\{\mathbf{v}_a^{(l)} | l = [0, 1, \cdots L]\}). \quad (2)$$

B. Markov Logic Network

Markov logic networks (MLNs) combine first-order logic rules with probabilistic graphical models. In our SLHRec, we incorporate the MLN within the logic-aware module. To be specific, a MLN [49] is defined as follows:

Definition 1: Markov Logic Network (MLN). A MLN is a set of pairs (f_i, w_i), where f_i represents a rule in first-order logic and w_i is a real number (i.e., confidence score).

MLNs can be viewed as templates for constructing Markov networks [11]. It uses logic rules to define potential functions in graphical models. Each rule, expressed as $f_i(\cdot) : \mathcal{C} \times \cdots \times \mathcal{C} \rightarrow \{0, 1\}$, is a function formulated via the predicate combination over constant set $\mathcal{C}$. Furthermore, each rule f_i is associated with a real number w_i, which serves as a confidence score: a higher value implies a greater reliability of the rule.

C. Heterogeneous Information Network

In this work, we focus on HIN-based recommender systems. A HIN [37], [53], [59] is described as follows:

Definition 2: Heterogeneous Information Network (HIN). A HIN is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the node set and edge set, respectively. Each node $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ are associated with their respective type mapping functions: $\phi(v) : \mathcal{V} \rightarrow T_{\mathcal{V}}$ and $\psi(e) : \mathcal{E} \rightarrow T_{\mathcal{E}}$, where $T_{\mathcal{V}}$ and $T_{\mathcal{E}}$ represent node type set and edge type set, respectively, with the condition $|T_{\mathcal{V}}| + |T_{\mathcal{E}}| > 2$.

In the language of first-order logic, nodes and relations in HIN could be called constants and predicates [11], [47]. HIN $\mathcal{G}$ can be represented by a fact set $\mathcal{T}$ whose element is a triplet $t = (v_a, r_b, v_c)$, where $v_a \in \mathcal{V}$ and $v_c \in \mathcal{V}$ are nodes, and $r_b \in \mathcal{E}$ is the node relation. A binary indicator $\mathbb{I}_t$ is associated with each triplet to represent its label: $\mathbb{I}_t = 1$ if triplet t is observed in $\mathcal{G}$, and zero otherwise. We utilize $\mathcal{T}_{\mathcal{O}}$ and $\mathcal{T}_{\mathcal{H}}$ to denote the observed and hidden (unobserved) facts, respectively. In this paper, the open-world paradigm [79] is used, and unobserved facts are treated as latent variables.

D. Problem Definition

Considering a HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathcal{V}_U \subset \mathcal{V}$ and $\mathcal{V}_I \subset \mathcal{V}$ denote the user and item entity sets. An interaction matrix $\mathbf{Y}$ is associated with users and items, such that $\mathbf{Y} = \{y_{ui} = 1 : u \in \mathcal{V}_U, i \in \mathcal{V}_I, (u, i) \in \mathcal{E}\}$. The goal of our SLHRec is to use the interaction set $\mathbf{Y}$ and HIN $\mathcal{G}$ to predict user u's preference $\hat{y}_{ui}$ for item i that he has never engaged with before.

III. Methodology

Our approach SLHRec is composed of two key modules: the structure-aware module and logic-aware module. The former models the HIN at the structural level, capturing both local and non-local neighborhood information. The latter focuses on the logical relationships of the HIN. In addition, we develop cross-view contrastive learning to build a collaborative relation between two modules. During the inference phase, we integrate the outputs from both modules to generate recommendations. Figure 1 presents the framework structure of SLHRec. In what follows, we discuss each part of SLHRec.

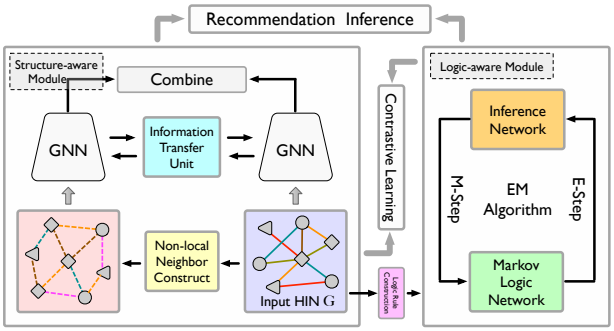

Fig. 1. Overall architecture of our proposed SLHRec.

A. Structure-aware Module

This module contains three components: (i) non-local neighborhood construction, which builds a non-local neighbor set for each node in the HIN; (ii) dual graph neural network, which refines node embeddings by modeling local and non-local neighborhood; and (iii) information transfer unit, which ensures the efficient interplay between two aggregations. In the following, we introduce these components.

*1) **Non-local Neighborhood Construction**:* As mentioned in the introduction section, HINs exhibit the heterophily property. The literature [80] presents two ideas to use this feature for graph learning: non-local neighbor extension [30], [32], [46] and GNN architecture refinement [20], [72]. Our model use the former idea to build non-local structural neighbors via the HIN graph topology exploration.

We develop a three-step method for the non-local neighborhood construction in the HIN, as shown in Figure 2.

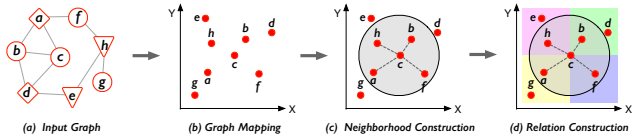

Fig. 2. Illustration of the non-local neighbor construction.

- Step 1. The first step maps the nodes of HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in a latent space. Specifically, we introduce a mapping function $m : v \rightarrow \mathbf{z}_v$, which transforms each node $v \in \mathcal{V}$ into a vector $\mathbf{z}_v \in \mathbb{R}^d$. This vector $\mathbf{z}_v$ symbolizes the node's coordinates in a d-dimensional latent space. During mapping, the HIN's structural and inherent properties are preserved and reflected as geometry relations in this space [46]. We employ Struc2Vec [48] as our embedding technique, noting that alternative embedding techniques [44], [56] could also be applied to infer the latent space.

- Step 2. Network embedding aids in preserving and transforming structural and topological patterns in the HIN into a more intuitive geometric representation in a latent space [12], [78]. In this space, nodes that share structural or semantic similarities with a target node are positioned closer, whereas nodes associated with irrelevant or noisy information are placed further away [46]. This spatial arrangement facilitates the creation of the non-local structural neighborhood based on the latent space. Specifically, for a given node $v \in \mathcal{V}$, we define its neighborhood as $\mathcal{N}_v^s$, consisting of the K nearest nodes to v in the latent space. Aggregating information over this neighborhood enables the capture of non-local dependencies in the HIN.
- Step 3. Following [46], we utilize a function φ to consider node relations in the latent space. Such a function takes an ordered pair $(\mathbf{z}_v, \mathbf{z}_u)$ of nodes v and u as inputs, and computes a geometric relation r from v to u:

$$\varphi : (\mathbf{z}_v, \mathbf{z}_u) \rightarrow r \in \mathcal{R}_s, \tag{3}$$

  where $\mathcal{R}_s$ is the geometric relation set in the latent space. Figure 2(d) shows an example in a 2-dimensional space, we categorize relations into set $\mathcal{R}_s$ comprising four types: upper-left, upper-right, lower-left, and lower-right.

The aforementioned processes serve a dual purpose. First, they allow us to extract non-local neighborhood information from the HIN. Second, they enable the modeling of geometric relations among neighborhoods, providing a more fine-grained understanding of the HIN structure.

*2) **Dual Graph Neural Network**:* We learn representations of nodes in HIN by considering their original and constructed non-local neighborhoods, as shown in Figure 3(a).

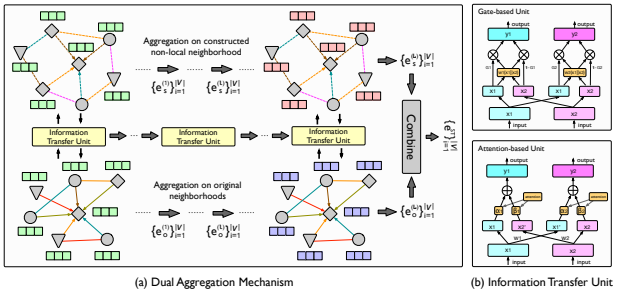

Fig. 3. Illustration of the (a) dual aggregation mechanism and (b) two types of information transfer units.

Specifically, we employ a dual aggregation mechanism to model neighborhoods through GNN principles. For a given node $v_a \in \mathcal{V}$, with its original neighborhood $\mathcal{N}_a^o$ in the HIN, and constructed non-local neighborhood $\mathcal{N}_a^s$, the dual mechanism in the l-th aggregation operates as follows:

$$\mathbf{e}_{o,a}^{(l)} = \text{Aggre}\big(\mathbf{e}_{o,b}^{(l-1)} | v_b \in \mathcal{N}_a^o\big), \tag{4}$$

$$\mathbf{e}_{s,a}^{(l)} = \text{Aggre}\big(\mathbf{e}_{s,b}^{(l-1)} | v_b \in \mathcal{N}_a^s\big), \tag{5}$$

where $\mathbf{e}_{o,a}^{(l)}$ and $\mathbf{e}_{s,a}^{(l)}$ represents the embeddings aggregated through two neighbor types of node v_a, and $\mathbf{e}_{o,a}^{(0)} = \mathbf{e}_{s,a}^{(0)} = \mathbf{e}_a$ is node v_a's initial embedding representation.

It is noted that each neighbor type is associated with multiple relations to v_a, i.e., HIN explicit relation $\mathcal{R}_o$ for original neighborhood $\mathcal{N}_a^o$ and geometric relation $\mathcal{R}_s$ for constructed non-local neighborhood $\mathcal{N}_a^s$. Thus, the Aggre function should effectively model these relation information. Here, we consider two ideas of neighborhood aggregations: bi-level aggregation and unified aggregation. For simplicity, we omit superscripts (or subscripts) s and o when introducing these methods.

The idea of bi-level aggregation [6], [50] is to first aggregate representations from neighbors connected by the same relation, and then aggregates representations across relations. We use two strategies based on the idea, as follows:

- Average-based strategy [50]. This strategy uses normalized sum for the neighbor aggregation, as follows:

$$\mathbf{e}_a^{(l)} = \sum_{r \in \mathcal{R}} \sum_{b \in \mathcal{N}_a^r} \frac{1}{|\mathcal{N}_a^r|} \mathbf{W}_r^{(l)} \mathbf{e}_b^{(l-1)}, \tag{6}$$

  where $\mathcal{N}_a^r$ denotes neighbors connected through the relation $r \in \mathcal{R}$, term $1/|\mathcal{N}_a^r|$ ensures normalization, and $\mathbf{W}_r^{(l)}$ is the weight parameter for relation r in the l-th aggregation.
- Attention-based strategy [6]. This strategy assigns weights to neighbors using an attention mechanism:

$$\mathbf{e}_a^{(l)} = \sum_{r \in \mathcal{R}} \sum_{b \in \mathcal{N}_a^r} \alpha_{a,b,r}^{(l)} \mathbf{W}_r^{(l)} \mathbf{e}_b^{(l-1)}, \tag{7}$$

  where α is the attention weight, and is calculated as:

$$\alpha_{a,b,r}^{(l)} = \frac{\exp\left(\mathbf{P}_r^{(l)} \mathbf{e}_a^{(l-1)} + \mathbf{Q}_r^{(l)} \mathbf{e}_b^{(l-1)}\right)}{\sum_{r' \in \mathcal{R}} \sum_{b' \in \mathcal{N}_a^{r'}} \exp\left(\mathbf{P}_{r'}^{(l)} \mathbf{e}_a^{(l-1)} + \mathbf{Q}_{r'}^{(l)} \mathbf{e}_{b'}^{(l-1)}\right)}, \tag{8}$$

  where $\mathbf{P}_r^{(l)}$ and $\mathbf{Q}_r^{(l)}$ are the attention parameters associated with relation r during the l-th aggregation iteration.

Unlike the bi-level aggregation, unified aggregation [58], [60] considers all relation types in one aggregation. We use two strategies based on this idea [58], [60], as follows:

- Average-based strategy. Normalized sum is used for aggregating neighbor features, as follows:

$$\mathbf{e}_a^{(l)} = \sum_{b \in \mathcal{N}_a} \frac{1}{|\mathcal{N}_a|} \mathbf{W}_r^{(l)} \mathbf{e}_b^{(l-1)}, \tag{9}$$

  where $\mathcal{N}_a$ is neighbors for node v_a.
- Attention-based strategy. This strategy assigns weights to neighbors using an attention mechanism, as follows:

$$\mathbf{e}_a^{(l)} = \sum_{b \in \mathcal{N}_a} \alpha_{a,b}^{(l)} \mathbf{W}_r^{(l)} \mathbf{e}_b^{(l-1)}, \tag{10}$$

  where attention weight α is calculated as follows:

$$\alpha_{a,b}^{(l)} = \frac{\exp\left(\mathbf{P}_r^{(l)} (\mathbf{e}_a^{(l-1)} || \mathbf{e}_b^{(l-1)})\right)}{\sum_{b' \in \mathcal{N}_a} \exp\left(\mathbf{P}_r^{(l)} (\mathbf{e}_a^{(l-1)} || \mathbf{e}_{b'}^{(l-1)})\right)}, \tag{11}$$

  where $||$ denotes the concatenation operation.

We compare the performance of these operations in Subsection IV-E. It is important to emphasize that our dual aggregation mechanism is model-agnostic. This characteristic allows

for its application alongside various existing methods [22], [37], [73], in addition to the above aggregation operations.

*3) **Information Transfer Unit**:* The dual aggregation mechanism incorporates two types of neighbor information in the HIN. However, it overlooks the potential interplay between aggregations. We posit that the two aggregations are not independent but interrelated, each providing a unique perspective on the HIN structure. To capture the interactions between aggregations, inspired by [29], we introduce an information transfer unit (as shown in Figure 3(b)), denoted as $\mathcal{K}$.

Without loss of generality, the input of unit $\mathcal{K}$ is represented as an ordered pair $(\mathbf{x}_1, \mathbf{x}_2)$, and the corresponding outputs are denoted by $(\mathbf{y}_1, \mathbf{y}_2)$. Based on this, unit $\mathcal{K}$ is defined as:

$$(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{K}(\mathbf{x}_1, \mathbf{x}_2). \tag{12}$$

To reference either output of $\mathcal{K}$, we use the notation $[x_i]$:

$$\mathbf{y}_i = \mathcal{K}(\mathbf{x}_1, \mathbf{x}_2)[x_i], \quad i \in \{1, 2\}. \tag{13}$$

Specifically, we designed two types of information transfer units, one based on the gating mechanism and the other based on the attention mechanism. They are defined as follows:

- Gated-based unit. This unit devises a gating mechanism to transfer information between embeddings, as follows:

$$\mathbf{y}_1 = \mathrm{G}_1 \odot \mathbf{x}_2 + (\mathbf{1} - \mathrm{G}_1) \odot \mathbf{x}_1, \tag{14}$$
$$\mathbf{y}_2 = \mathrm{G}_2 \odot \mathbf{x}_1 + (\mathbf{1} - \mathrm{G}_2) \odot \mathbf{x}_2, \tag{15}$$

  where G_1, G_2 are gated structures, which are defined as:

$$\mathrm{G}_1 = \sigma\big(\mathbf{w}_{g_1}(\mathbf{x}_1 || \mathbf{x}_2)\big), \tag{16}$$
$$\mathrm{G}_2 = \sigma\big(\mathbf{w}_{g_2}(\mathbf{x}_1 || \mathbf{x}_2)\big), \tag{17}$$

  where $\mathbf{w}_\cdot$ is the weight matrix and σ is the sigmoid function.
- Attention-based unit [29]. This unit uses an attention mechanism to transfer information between embeddings:

$$\mathbf{y}_1 = \alpha_1 \cdot \mathbf{x}_2 + \beta_1 \cdot \mathbf{x}_1, \tag{18}$$
$$\mathbf{y}_2 = \alpha_2 \cdot \mathbf{x}_1 + \beta_2 \cdot \mathbf{x}_2, \tag{19}$$

  where $\alpha_\cdot$, $\beta_\cdot$ are attention weights, which are defined as:

$$\alpha_1 = \frac{\exp(\mathbf{w}_{11}\mathbf{x}_1)}{\exp(\mathbf{w}_{11}\mathbf{x}_1) + \exp(\mathbf{w}_{12}\mathbf{x}_2)} = 1 - \beta_1 \tag{20}$$
$$\alpha_2 = \frac{\exp(\mathbf{w}_{21}\mathbf{x}_2)}{\exp(\mathbf{w}_{21}\mathbf{x}_2) + \exp(\mathbf{w}_{22}\mathbf{x}_1)} = 1 - \beta_2 \tag{21}$$

  where $\mathbf{w}_{\cdot\cdot}$ is the weight matrix of the attention mechanism.

This information transfer unit integrates with the original dual aggregation mechanism (i.e., Equations (4) and (5)), as:

$$\mathbf{e}_{o,a}^{(l)} = \mathrm{Aggre}\big(\mathcal{K}^{(l)}(\mathbf{e}_{o,b}^{(l-1)}, \mathbf{e}_{s,b}^{(l-1)})[o] | v_b \in \mathcal{N}_a^o\big), \tag{22}$$
$$\mathbf{e}_{s,a}^{(l)} = \mathrm{Aggre}\big(\mathcal{K}^{(l)}(\mathbf{e}_{o,b}^{(l-1)}, \mathbf{e}_{s,b}^{(l-1)})[s] | v_b \in \mathcal{N}_a^s\big), \tag{23}$$

By incorporating the information transfer unit, we facilitate the exchange of relevant information between the two aggregations, enhancing their integration. The efficacy of this unit will be evaluated in the experiments (cf. Subsection IV-E).

*4) **Prediction and Optimization**:* After L layers of aggregation, we combine the embeddings from each layer (including the initial embeddings) to form the structure-aware representation $\mathbf{e}_a^{\mathrm{ST}}$ for node v_a, as follows:

$$\mathbf{e}_{o,a}^* = \mathrm{Readout}\big(\mathbf{e}_{o,a}^{(l)} | l = [0, 1, \cdots L]\}\big), \tag{24}$$
$$\mathbf{e}_{s,a}^* = \mathrm{Readout}\big(\mathbf{e}_{s,a}^{(l)} | l = [0, 1, \cdots L]\}\big), \tag{25}$$
$$\mathbf{e}_a^{\mathrm{ST}} = \mathrm{Combine}\big(\mathbf{e}_{o,a}^*, \mathbf{e}_{s,a}^*\big), \tag{26}$$

where function Readout is defined as the average summation operation, and function Combine is used to combine $\mathbf{e}_{o,a}^*$ and $\mathbf{e}_{s,a}^*$, which first concatenates them and then processes the combined representation through a multi-layer perceptron.

Thus, given a user v_u and an item v_i, v_u's preference for the v_i in structure-aware module is modeled as:

$$\hat{y}_{ui}^{\mathrm{ST}} = p(\mathbf{e}_{v_u}^{\mathrm{ST}}, \mathbf{e}_{v_i}^{\mathrm{ST}}), \tag{27}$$

where p is the prediction function, defined as the inner product.

Based on the above prediction, we use the cross-entropy loss function [33], [58] to optimize the structure-aware module:

$$\mathcal{L}_{\text{structure}} = -\sum\nolimits_{v_u \in \mathcal{V}_U} \Big(\sum\nolimits_{v_i : y_{ui}=1} \mathcal{J}\big(y_{ui}, \hat{y}_{ui}^{\mathrm{ST}}\big) - \sum\nolimits_{j=1}^{C_u} \mathbb{E}_{v_j \sim \mathcal{P}_{v_j}} \mathcal{J}\big(y_{uj}, \hat{y}_{uj}^{\mathrm{ST}}\big) \Big), \tag{28}$$

where $\mathcal{J}$ is the cross-entropy function, and $\mathcal{P}$ is the negative sampling distribution. For each user v_u, $C_u = |v_j : y_{uj} = 1|$ is the number of negative samples.

B. Logic-aware Module

This module is used to model the logic relations in the HIN. An overview of this module is shown in Figure 4.

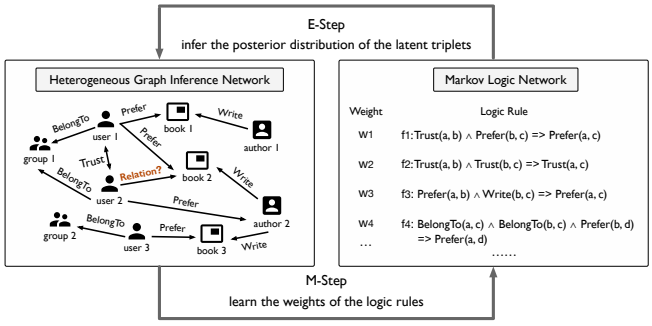

Fig. 4. Illustration of the logic-aware module.

Logic-aware module uses a MLN [49], [79] with logic rules to define the joint distribution of all possible facts in the HIN. The MLN is trained utilizing a variational Expectation-Maximization (EM) algorithm [42], [47], which contains expectation (E-step) and maximization (M-step) phases. During the E-step, a heterogeneous graph inference network (HGIN) is introduced to infer missing facts. This process ensures that the knowledge of the logic rules is integrated into the node representations. In the M-step, the logic rules' weights are learned by considering both observed facts and those inferred by the HGIN. This interplay provides meaningful supervisory signals that enhance the process of representation learning. In the following, we first introduce the process of tailoring the

MLN for HINs. We then detail the training phase by utilizing the variational EM algorithm. Finally, we present the design of the inference network HGIN.

*1) **Variational EM for MLN**:* The purpose of the logic-aware module is to infer the labels for the hidden triplets $\mathcal{T}_{\mathcal{H}}$ based on the observed triplets $\mathcal{T}_{\mathcal{O}}$ [11], [47], [79].

We first construct logic rules related to the HIN based on its contained information. For details on the rule construction process, please see Section IV-A1. With a set of logic rules $\mathcal{F}$, we use the MLN to define the joint probability distribution of both the observed and hidden triplets [47]:

$$\begin{aligned} P_w(\mathcal{T}_{\mathcal{O}}, \mathcal{T}_{\mathcal{H}}) &= \frac{1}{Z(w)} \exp\Big(\sum\nolimits_{f \in \mathcal{F}} w_f \sum\nolimits_{g \in G_f} \mathbb{1}\{g \text{ is true}\} \Big) \\ &= \frac{1}{Z(w)} \exp\Big(\sum\nolimits_{f \in \mathcal{F}} w_f n_f(\mathcal{T}_{\mathcal{O}}, \mathcal{T}_{\mathcal{H}}) \Big), \end{aligned} \tag{29}$$

where $Z(w)$ is the partition function, w_f is the weight assigned to each rule $f \in \mathcal{F}$, G_f is a ground rule set by instantiating the placeholders in rule f with nodes in the HIN, and $n_f(\mathcal{T}_{\mathcal{O}}, \mathcal{T}_{\mathcal{H}})$ denotes the number of the rule f's true groundings based on the observed triplets $\mathcal{T}_{\mathcal{O}}$ and hidden triplets $\mathcal{T}_{\mathcal{H}}$.

This model can be optimized by maximizing log-likelihood $\log P_w(\mathcal{T}_{\mathcal{O}})$ for observed triplets $\mathcal{T}_{\mathcal{O}}$ [79]. However, direct optimization is impractical due to the computational complexity of integrating over all hidden triplets $\mathcal{T}_{\mathcal{H}}$. To circumvent this issue, we focus on maximizing the Evidence Lower BOund (ELBO) of the log-likelihood [47], [79], as follows:

$$\begin{aligned} &\log P_w(\mathcal{T}_{\mathcal{O}}) \geq \mathcal{L}_{ELBO}(Q_\theta, P_w) \\ &= \mathbb{E}_{Q_\theta(\mathcal{T}_{\mathcal{H}})}\big[\log P_w(\mathcal{T}_{\mathcal{O}}, \mathcal{T}_{\mathcal{H}})\big] - \mathbb{E}_{Q_\theta(\mathcal{T}_{\mathcal{H}})}\big[\log Q_\theta(\mathcal{T}_{\mathcal{H}})\big], \end{aligned} \tag{30}$$

in which $Q_\theta(\mathcal{T}_{\mathcal{H}})$ denotes the variational distribution for the hidden triplets. The above inequality becomes an equality when $Q_\theta(\mathcal{T}_{\mathcal{H}})$ is equivalent to the true posterior distribution $P_w(\mathcal{T}_{\mathcal{H}}|\mathcal{T}_{\mathcal{O}})$. We can efficiently maximize this lower bound using the EM algorithm [42], which alternates between an E-step for inference and a M-step for parameter learning. In the following, we introduce these two steps.

*2) **E-step**:* During the E-step, we infer the posterior distribution of hidden triplets, fixing P_w while adjusting Q_θ to minimize the KL divergence between $Q_\theta(\mathcal{T}_{\mathcal{H}})$ and $P_w(\mathcal{T}_{\mathcal{H}}|\mathcal{T}_{\mathcal{O}})$. The true posterior is approximated by a mean-field distribution. Specifically, $Q_\theta(\mathcal{T}_{\mathcal{H}})$ is inferred as follows:

$$Q_\theta(\mathcal{T}_{\mathcal{H}}) = \prod\nolimits_{t \in \mathcal{T}_{\mathcal{H}}} Q_\theta(\mathbb{I}_t), \tag{31}$$

where $\mathbb{I}_t$ is the label indicator for t, $Q_\theta(\mathbb{I}_t)$ is modeled as the Bernoulli distribution. We use a heterogeneous graph inference network to parameterize the variational posterior Q_θ, which will be detailed in Subsection III-B4.

The first term in Equation (30) involves the summation operation, which introduces computational complexity. To this end, we use the ways in [47], [79] and implement a mini-batch strategy to manage the exponential number of summations. In each iteration, a batch of grounding rules is sampled. For every rule in this batch, the first term of Equation (30) is calculated by taking the expectation of corresponding potential function, using the posterior of the associated latent variables. This mean-field approximation allows for the breakdown of the overall expectation across the MLN into local expectations for each ground rule [79]. For the second term in Equation (30), the local entropy is computed using the posteriors of the latent variables in the current batch. This strategy ensures that the computation remains tractable and efficient.

Moreover, we follow the methods in [47], [79] and use the labeled triplets in $\mathcal{T}_{\mathcal{O}}$ to refine the inference network:

$$\mathcal{L}_{LABEL}(Q_\theta) = \sum\nolimits_{t \in \mathcal{T}_{\mathcal{O}}} \log Q_\theta(\mathbb{I}_t). \tag{32}$$

This operation is complementary to the ELBO.

Thus, the comprehensive objective function is:

$$\mathcal{L}_\theta = \mathcal{L}_{ELBO}(Q_\theta, P_w) + \mathcal{L}_{LABEL}(Q_\theta). \tag{33}$$

This function considers both the knowledge in logical rules and supervision signals from labeled data.

*3) **M-step**:* During the M-step (i.e., learning step), we aim to learn the weights of the logic rules in the MLN. In this phase, Q_θ is fixed while P_w is adjusted to maximize the log-likelihood of the data, i.e., $\mathbb{E}_{Q_\theta(\mathcal{T}_{\mathcal{H}})}\big[\log P_w(\mathcal{T}_{\mathcal{O}}, \mathcal{T}_{\mathcal{H}})\big]$. However, directly optimizing this log-likelihood is complex due to the involvement of the partition function $Z(w)$. To this end, we choose to maximize the PseudoLikelihood function [5], [47], [79], as follows:

$$\mathcal{L}_w = \mathbb{E}_{Q_\theta(\mathcal{T}_{\mathcal{H}})}\big[\sum\nolimits_{t \in \mathcal{T}_{\mathcal{H}}} \log P_w\big(\mathbb{I}_t|\mathbb{I}_{M(t)}\big)\big], \tag{34}$$

where $M(t)$ denotes the Markov blanket of t, including triplets that co-occur with t in the grounding of logic rules.

For rule f that connects t to its Markov blanket, we optimize its weight w_f by gradient descent, with the derivative [79]:

$$\nabla_{w_f} \mathbb{E}_{Q_\theta(\mathcal{T}_{\mathcal{H}})}\big[\log P_w\big(\mathbb{I}_t|\mathbb{I}_{M(t)}\big)\big] \simeq y_t - P_w\big(\mathbb{I}_t|\mathbb{I}_{M(t)}\big), \tag{35}$$

where $y_t = 1$ if t is an observed triplet and $y_t = Q_\theta(\mathbb{I}_t)$ if t is a hidden one. Leveraging the independence property of the MLN, we can efficiently compute the gradients of the weights based on the Markov blanket of each variable [79].

*4) **Inference Network**:* In the variational EM framework, the inference network (i.e., posterior model) is crucial. In our approach, we employ a heterogeneous graph inference network (HGIN) for this purpose. Specifically, distribution $Q_\theta(\mathcal{T}_{\mathcal{H}})$ in Equation (31) is defined as follows:

$$Q_\theta(\mathcal{T}_{\mathcal{H}}) = \prod\nolimits_{t \in \mathcal{T}_{\mathcal{H}}} \text{Ber}(\mathbb{I}_t|\text{HGIN}(v_a, r_b, v_c)), \tag{36}$$

where Ber represents the Bernoulli distribution.

The HGIN, which infers the plausibility (i.e., predicted probability) of a given triplet $t = \{v_a, r_b, v_c\}$, is composed of two components: an encoder and a decoder. The encoder uses the idea in [3] and calculates the distance between node v_a and node v_c based on their relationship r_b, as follows:

$$\text{dist}_t = \|\mathbf{W}_b \mathbf{e}_a^{\text{LO}} - (\mathbf{e}_c^{\text{LO}} + \mathbf{r}_b)\|_{\text{F}}^2, \tag{37}$$

where $\mathbf{W}_b$ and $\mathbf{r}_b$ are the weight matrix and embedding for relation r_b, respectively. Embeddings $\mathbf{e}_a^{\text{LO}}$ and $\mathbf{e}_c^{\text{LO}}$ are derived

from the initial node embeddings $\mathbf{e}_a$ and $\mathbf{e}_c$ using the linear transformation with weight parameter $\mathbf{W}_l$, as follows:

$$\mathbf{e}_a^{\text{LO}} = \mathbf{W}_l \mathbf{e}_a, \quad \mathbf{e}_c^{\text{LO}} = \mathbf{W}_l \mathbf{e}_c, \tag{38}$$

The decoder of HGIN, implemented using the Fermi–Dirac decoding approach [27], [44], utilizes the calculated distance dist_t to predict the probability of triplet t, as follows:

$$Q_\theta(\mathbb{I}_t) = \left(\exp\left(\frac{\text{dist}_t - \omega}{\varrho} \right) + 1 \right)^{-1}, \tag{39}$$

in which ω and ϱ are hyperparameters.

C. Cross-view Contrastive Learning

In contrastive learning, we enhance the model performance by aligning the outputs from the structure-aware and logic-aware modules, which provides supplementary supervisory signals. We regard the two module outputs as separate views, considering the representations of the same node v_i in different views (i.e., $\mathbf{e}_i^{\text{ST}}, \mathbf{e}_i^{\text{LO}}$) as positive pairs and those of different nodes as negative pairs. This strategy refines discriminative representations by contrasting positive and negative samples. We define our contrastive loss using the InfoNCE [45]:

$$\mathcal{L}_{\text{cl}} = \sum_{v_i \in \mathcal{V}} -\log \frac{\exp\left(c\big(\mathbf{e}_i^{\text{ST}}, \mathbf{e}_i^{\text{LO}}\big)/\tau \right)}{\sum_{v_{i'} \in \mathcal{V}} \exp\left(c\big(\mathbf{e}_i^{\text{ST}}, \mathbf{e}_{i'}^{\text{LO}}\big)/\tau \right)}, \tag{40}$$

where $c(\cdot, \cdot)$ denotes cosine similarity and τ is the hyperparameter which adjusts the scale of softmax function.

Our cross-view contrastive learning task refines discriminative representations by leveraging node embeddings from the structure-aware and logic-aware modules. By aligning these embeddings for the same node across views, we distill the distinctive information of each view into the other, thus improving both representations [38], [66]. We will study the impact of our contrastive learning in Subsection IV-E.

D. Model Optimization and Inference

In this subsection, we detail the optimization and inference process of our proposed method SLHRec.

*1) **Optimization**:* The optimization of SLHRec mainly contains structure-aware loss $\mathcal{L}_{\text{structure}}$, and logic-aware loss $\mathcal{L}_{\text{logic}}$. We employ a multi-task learning strategy for optimization:

$$\mathcal{L} = \mathcal{L}_{\text{structure}} + \mathcal{L}_{\text{logic}} + \lambda_1 \mathcal{L}_{\text{cl}} + \lambda_2 ||\Theta||_{\text{F}}^2, \tag{41}$$

where $\mathcal{L}_{\text{logic}}$ contains $\mathcal{L}_\theta$ and $\mathcal{L}_w$; λ_1, λ_2 adjust the contrastive and regularization strengths; Θ denotes model parameters.

*2) **Inference**:* For the inference phase, our SLHRec utilizes the structure-aware and logic-aware modules to learn user preferences for items. Specifically, given a user $v_u \in \mathcal{V}_U$ and an item $v_i \in \mathcal{V}_I$, v_u's preference for v_i is calculated as:

$$\hat{y}_{uv} = \alpha \cdot p(\mathbf{e}_{v_u}^{\text{ST}}, \mathbf{e}_{v_i}^{\text{ST}}) + (1 - \alpha) \cdot \text{HGIN}(v_u, r_p, v_i), \tag{42}$$

where α is a balancing factor between 0 and 1, and r_p denotes the interaction relations in triplet (v_u, r_p, v_i).

IV. Experiments

In this section, we first introduce the experimental settings. We then show the effectiveness of our SLHRec by comparing it with baselines. Next, a comprehensive analysis of SLHRec's components and hyper-parameters is conducted. Lastly, we explain the application of logic rules for reasoning in SLHRec.

A. Experimental Setup

This subsection introduces the datasets, baselines, hyper-parameter settings, and evaluation metrics.

*1) **Datasets**:* We utilize four real-world datasets: Douban[1], HetRec[2], Yelp[3], and KKBox[4]. (i) Douban dataset is for book recommendation includes user social relations and group affiliations, and book metadata such as author and publisher details; (ii) HetRec dataset, serving movie recommendation, contains actor, director, and tag details for movies; (iii) Yelp dataset is for business recommendation, including user social relations and business specifics like city and category; and (iv) KKBox dataset is for music recommendation, providing artist, lyricist, composer, and genre information for music.

For each dataset, we consider four types of logic rules [47]:

- Symmetric rules: A symmetric relation r implies that for two entities v_x and v_y, if v_x is related to v_y via r, then v_y is also related to v_x by the same relation. This can be expressed as $\forall v_x, v_y \in \mathcal{V}, (v_x, r, v_y) \Rightarrow (v_y, r, v_x)$.
- Inverse rules: When a relation r_j serves as the inverse of r_i, it means that if an entity v_x is related to another entity v_y through r_i, then v_y is related to v_x through r_j. This can be expressed as $\forall v_x, v_y \in \mathcal{V}, (v_x, r_i, v_y) \Rightarrow (v_y, r_j, v_x)$.
- Subrelation rules: If relation r_j is a subrelation of r_i, this means that the presence of r_i between any two entities v_x and v_y establishes r_j between them as well. This can be expressed as $\forall v_x, v_y \in \mathcal{V}, (v_x, r_i, v_y) \Rightarrow (v_x, r_j, v_y)$.
- Composition rules: A relation r_k being a composition of r_i and r_j means that for any three entities v_x, v_y, v_z, if v_x is related to v_y via r_i and v_y is related to v_z via r_j, there exists a relation r_k between v_x and v_z. This can be expressed as $\forall v_x, v_y, v_z \in \mathcal{V}, (v_x, r_i, v_y) \wedge (v_y, r_j, v_z) \Rightarrow (v_x, r_k, v_z)$.

According to the four types of rules and metadata contained in datasets, we hand-code logic rules for each dataset. Table II presents the statistical information of four datasets.

TABLE II
DATASET STATISTICS.

Datasets	# users	# items	# interactions	# facts	# rules
Douban	11,951	20,521	264,735	593,847	42
HetRec	2,112	8,665	376,105	526,652	35
Yelp	50,937	23,217	801,632	2,078,987	24
KKBox	17,086	13,585	1,174,028	1,564,771	54

[1]http://book.douban.com
[2]https://grouplens.org
[3]http://www.yelp.com
[4]https://www.kkbox.com

*2) **Baselines**:* To evaluate our model, we compare SLHRec with state-of-the-art methods from various research lines, covering (i) feature-enhanced methods (AFM [69] and DeepFM [17]), (ii) metapath-based methods (HERec [52] and MCRec [21]), (iii) heterogeneous graph convolution methods (HGT [22] and SimpleHGN [37]), (iv) logic-enhanced methods (ExpressGNN [79] and pGAT [19]), and (v) heterogeneous graph self-supervised learning methods (HeCo [63] and HGCL [9]). We briefly introduce these approaches below:

- **AFM** improves traditional factorization machines by using an attention network [69]. We concatenate the user, item, and side information embeddings to form the input.
- **DeepFM** merges factorization machines and deep learning, facilitating low- and high-order feature interactions [17]. We provide the same inputs as AFM for DeepFM.
- **HERec** employs a meta-path based random walk for network embedding, with embeddings refined by fusion functions and integrated into a matrix factorization model [52].
- **MCRec** uses a co-attention mechanism for HIN-based recommendation by representing meta-paths and considering their mutual effect with user-item pairs [21].
- **HGT** adapts GNNs for heterogeneous graphs by introducing node- and edge-type parameters, enabling meaningful representations for diverse node and edge types [22].
- **SimpleHGN** utilizes the idea of graph attention networks, designing edge-type embeddings, residual connections, and L2 normalization for heterogeneous graph modeling [37].
- **ExpressGNN** utilizes MLNs for modeling heterogeneous graphs and uses GNNs as the inference network [79].
- **pGAT** uses MLNs for modeling heterogeneous graphs and it uses a graph attention network for inference [19].
- **HeCo** leverages the cross-view contrastive learning task from network schema and meta-path views to capture HIN local and high-order structures [63].
- **HGCL** integrates heterogeneous relational semantics with user-item interactions by knowledge transformers and utilizes meta networks for contrastive augmentation [9].

*3) **Hyper-parameter Settings**:* We split each dataset into training, validation, and test sets with the ratio of 6:2:2. The Adam optimizer is leveraged for optimizing all approaches. We explore the learning rates within the set $\{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}\}$ and batch sizes among $\{128, 256, 512, 1024, 2048\}$. The embedding size is selected from values of $\{8, 16, 32, 64, 128\}$ for baselines. For SLHRec, we set embedding size as 32. In the non-local neighborhood construction, we use four-fold geometrical relations and set the non-local neighbor size as 5. In the dual aggregation mechanism, we use the bi-level average-based aggregation and gated-based information transfer unit by default. Additionally, we set the aggregation layer as 2. For the inference balancing factor α, it is searched in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. We will study the impact of several key hyper-parameters in Section IV-F.

*4) **Evaluation Metrics**:* Click-through rate prediction and top-K recommendation tasks are used for evaluation. For the former, we use the AUC and F1 as the metrics. In the latter case, we use precision (P@K) and recall (R@K) as the metrics, and define K as 10. Following [21], [28], [33], for each user, we sample 2000 negative items and combine these items with positive items in ranking to reduce the computational cost.

B. Empirical Study

As mentioned in the introduction, HINs exhibit the heterophily property. This empirical study is utilized to show this property of HINs. Following [46], [80], [81], we employ metrics, H_{node} and H_{edge}, to measure heterophily and homophily in HINs. H_{node} represents the average proportion of each node's neighbors belonging to the same class, and H_{edge} measures the proportion of edges connecting nodes of the same class. These metrics are defined as follows:

$$\mathrm{H}_{node} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\frac{|\{u\in\mathcal{N}_v^o : l_v = l_u\}|}{|\mathcal{N}_v^o|}, \tag{43}$$

$$\mathrm{H}_{edge} = \frac{|\{(v,u)\in\mathcal{E} : l_v = l_u\}|}{|\mathcal{E}|}, \tag{44}$$

where $l.$ is the node class and $\mathcal{N}_v^o$ is the neighborhood of node v in the HIN. The values of H_{node} and H_{edge} range from 0 to 1. Graphs exhibiting strong homophily will have H_{node} and H_{edge} values close to 1, whereas those with strong heterophily will have values closer to 0. Using these metrics, we analyze the four datasets in our experiment. The results in Table III show that all datasets have the heterophily property.

TABLE III
EMPIRICAL STUDY ON THE FOUR DATASETS.

Datasets	Douban	HetRec	Yelp	KKBox
H_{node}	0.0721	0.0597	0.0374	0.0407
H_{edge}	0.2252	0.0633	0.1447	0.0445

C. Performance Comparison

Table IV reports the comparison results of the CTR prediction and top-K recommendation tasks. The key findings are as follows: (i) MCRec outperforms HERec, underscoring the significance of deep learning in HIN-based recommender systems. (ii) Incorporating graph structural information enhances performance. For instance, graph-based recommenders generally outperform methods such as AFM, DeepFM, HERec, and MCRec, highlighting the value of recognizing entity relations in HINs. (iii) Self-supervised approaches such as HeCo and HGCL show highly competitive performance, indicating the effectiveness of extracting self-supervised signals in HINs. (iv) Our proposed method SLHRec shows superior performance. Specifically, in the CTR prediction, SLHRec achieves average AUC gains of 4.33%, 1.76%, 3.58%, and 3.49% across the four datasets. In the top-K recommendation, SLHRec outperforms the best baselines with increases in R@10 by 3.62%, 6.51%, 11.81%, and 7.15% on the four datasets.

D. Performance w.r.t Sparsity Degrees

This subsection evaluates the performance of our SLHRec in handling data sparsity. To this end, we vary the ratio of the

TABLE IV
PERFORMANCE COMPARISON RESULTS OF CLICK-THROUGH RATE PREDICTION AND TOP-K RECOMMENDATION TASKS.

Method	Douban				HetRec				Yelp				KKBox			
	AUC	F1	P@10	R@10	AUC	F1	P@10	R@10	AUC	F1	P@10	R@10	AUC	F1	P@10	R@10
AFM	0.7683	0.6975	0.0791	0.1650	0.8686	0.7934	0.3470	0.1454	0.9003	0.8281	0.0412	0.1176	0.8878	0.8087	0.2620	0.2153
DeepFM	0.7700	0.6948	0.0782	0.1698	0.8713	0.7987	0.3396	0.1401	0.9057	0.8386	0.0399	0.1148	0.8941	0.8128	0.2522	0.2058
HERec	0.7424	0.6706	0.0704	0.1569	0.8579	0.7818	0.3327	0.1352	0.8622	0.7904	0.0260	0.0989	0.8600	0.7881	0.2354	0.1931
MCRec	0.7619	0.6951	0.0776	0.1645	0.8670	0.7955	0.3363	0.1408	0.8892	0.8301	0.0361	0.1051	0.8765	0.7959	0.2495	0.2050
HGT	0.7788	0.7113	0.0820	0.1735	0.8717	0.7991	0.3385	0.1418	0.9106	0.8445	0.0434	0.1244	0.8899	0.8115	0.2591	0.2132
SimpleHGN	0.7732	0.6969	0.0825	0.1754	0.8802	0.7975	0.3442	0.1437	0.9115	0.8458	0.0426	0.1213	0.9073	0.8260	0.2815	0.2388
ExpressGNN	0.7865	0.7134	0.0853	0.1808	0.8729	0.8008	0.3445	0.1450	0.9098	0.8461	0.0445	0.1282	0.9022	0.8214	0.2554	0.2100
pGAT	0.7829	0.7127	0.0847	0.1781	0.8714	0.8003	0.3498	0.1476	0.9039	0.8377	0.0423	0.1237	0.8998	0.8219	0.2709	0.2205
HeCo	0.7822	0.7040	0.0808	0.1723	0.8821	0.7994	0.3516	0.1467	0.9150	0.8503	0.0443	0.1290	0.9019	0.8222	0.2603	0.2145
HGCL	0.7801	0.7068	0.0831	0.1772	0.8836	0.8020	0.3544	0.1506	0.9104	0.8400	0.0467	0.1329	0.9057	0.8280	0.2809	0.2357
SLHRec	**0.8058**	**0.7357**	**0.0892**	**0.1876**	**0.8880**	**0.8066**	**0.3837**	**0.1611**	**0.9339**	**0.8757**	**0.0520**	**0.1507**	**0.9236**	**0.8505**	**0.3018**	**0.2572**

training set from 80% to 20%, while keeping the validation and test sets unchanged. We conduct experiments using three datasets, and compare the performance of our SLHRec against several strong baselines, including SimpleGNN, ExpressGNN, and HGCL. The AUC results are shown in Figure 5. We find that as the training set ratio decreases, the performance of all methods decrease. This trend emphasizes the necessity of plentiful user-item interactions for high-quality recommendations. Furthermore, when the ratio is reduced to 20%, our SLHRec outperforms baselines, which shows the robustness and effectiveness of SLHRec in sparsity scenarios.

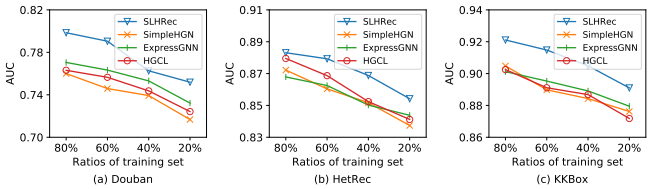

Fig. 5. The AUC results on three datasets with different training set ratios.

E. *Detailed Model Analysis*

This subsection studies the effect three components in our SLHRec: the structure-aware module, the logic-aware module, and the cross-view contrastive learning.

*1) **Effect of the Structure-aware Module**:* We first study the effect of the dual aggregation mechanism in the structure-aware module. We consider several operations, which can allow us to analyze the effect of the non-local neighborhood and the proposed aggregation backbones, as follows:

- w/o NLN: This operation removes the non-local neighborhood aggregation in the dual graph neural network.
- B-AVE: This operation uses bi-level average-based methods, which is leveraged by default in other experiments
- B-ATT: This operation uses bi-level attention-based methods.
- U-AVE: This operation uses unified average-based methods.
- U-ATT: This operation uses unified attention-based methods.

Table V presents the AUC results for the above operations. We find that four SLHRec variants that consider non-local neighborhoods outperform SLHRec-(w/o NLN), demonstrating that incorporating constructed non-local neighborhood information enhances performance. In addition, the performance of the four aggregation strategies varies across datasets. Thus, the adoption of different operations is context-dependent.

TABLE V
EFFECT OF THE DUAL AGGREGATION MECHANISM

Models	Douban	HetRec	Yelp	KKBox
w/o NLN	0.7954	0.8850	0.9198	0.9155
B-AVE	0.8058	0.8880	0.9339	0.9236
B-ATT	0.8060	0.8892	0.9331	0.9289
U-AVE	0.8076	0.8905	0.9325	0.9224
U-ATT	0.8073	0.8887	0.9350	0.9271

We further investigate the effect of our information transfer units. Specifically, we consider three operations as:

- w/o ITU: This operation removes the transfer units.
- Gate: This operation uses gated-based units.
- Attention: This operation uses attention-based units.

Table VI presents the AUC results for these operations. We find that either using gated-based or attention-based units achieves better results than removing units, underscoring the importance of these units in facilitating effective knowledge and information sharing between dual aggregations. In addition, two units differently affect the performance in practice, which should be set carefully for different scenarios.

TABLE VI
EFFECT OF THE INFORMATION TRANSFER UNIT

Models	Douban	HetRec	Yelp	KKBox
w/o ITU	0.8002	0.8864	0.9273	0.9227
Gate	0.8058	0.8880	0.9339	0.9236
Attention	0.8040	0.8895	0.9312	0.9235

*2) **Effect of the Logic-aware Module**:* We study the effect of the logic-aware module from the following two perspectives. Firstly, we use logic rules with quantity ratios of 75%, 50%, and 25% during the training, which helps us evaluate

whether logic rules influence the performance. Secondly, we introduce a variant called SLHRec-M, which excludes the M-step in the EM algorithm of the logic-aware module. This allows us to investigate whether the weights of the logic rules can be updated to affect the model performance.

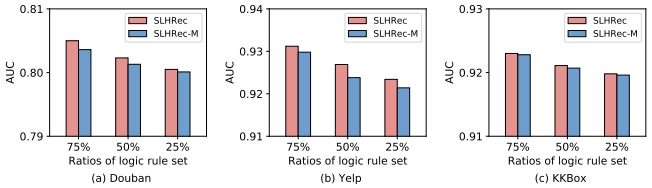

Fig. 6. The AUC results on three datasets with different logic rule ratios.

The AUC results on the three datasets are shown in Figure 6. From these results, the following observations could be made: (i) As the number of logic rules decreases, the model performance decreases, indicating that using logic rules contributes to the recommendation prediction. (ii) The combined use of both the variational E-Step and M-Step generally outperforms the use of only the E-Step. By learning the weights of logic rules, our model achieves a nice performance.

*3) **Effect of the Cross-view Contrastive Learning**:* This subsection studies the impact of cross-view contrastive learning by investigating three hyperparameters (i.e., strength λ_1, temperature τ, and batch size b). Figure 7 shows the impact of these hyperparameters on Douban and Yelp datasets. For brevity, only two datasets are presented and the result trends for HetRec and KKBox datasets are similar.

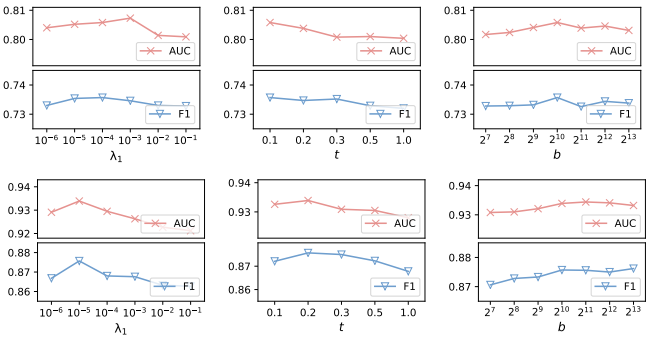

Fig. 7. Effect of self-supervised strength λ_1, temperature τ, and batch size b on Douban (in the top row) and Yelp (in the bottom row) datasets.

From these results, we can find that: (i) For self-supervised strength λ_1, an excessively high value can cause the model to prioritize self-supervised learning at the expense of performance, whereas too low a value may not distill sufficient self-supervised information. Optimal performance is generally achieved with λ_1 between 10^{-5} and 10^{-3}. (ii) The temperature τ plays a crucial role in the mining of hard negative samples [66]. Our findings suggest that a τ value of 0.1 or 0.2 yields a nice performance in most scenarios. A higher τ tends to degrade performance, which is consistent with prior research [34], [66]. (iii) Batch size b, essential for in-batch negative sampling, directly impacts learning. We test various batch sizes and observed an initial increase in accuracy, followed by stabilization. The default batch size is set to 1024, in line with common practices in implementations of baselines, and it shows effective performance in our method.

F. Hyperparameter Study

In Subsection IV-E, we study the hyperparameters related to contrastive learning. This subsection extends our analysis to additional hyperparameters (i.e., embedding size d, non-local neighbor size K, and prediction factor α) related to our model. The results for Douban and Yelp datasets are shown in Figure 8. The observations can be summarized as follows: (i) Embedding size d plays a crucial role in determining the representation ability. Increasing d initially enhances performance because a larger d encodes more information. However, if d becomes excessively large, it may lead to overfitting. Empirically, we find that when d is set to 32 or 64, SLHRec could achieve satisfactory accuracy. (ii) For non-local neighbor size K, we find that the model performance enhances with an increasing number of neighbors, showing the utility of neighbor construction in improving recommendations. However, performance declines when the neighbor number becomes excessive, suggesting susceptibility to the noise. (iii) As the prediction factor α is adjusted from 0.1 towards its optimal value, we observe an improvement in performance, followed by a decline. This trend suggests that the structure-aware module and logic-aware module are complementary. Moreover, the effect of α is dataset-dependent. For instance, setting α as 0.5 and 0.7 yields nice performances on Douban and HetRec datasets, respectively. Therefore, it is crucial to set α carefully according to the specific application.

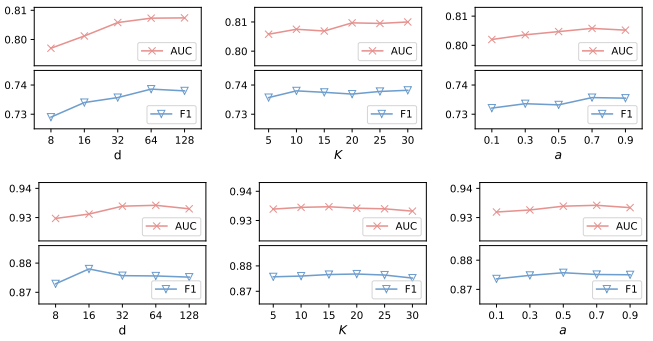

Fig. 8. Effect of embedding size d, non-local neighbor size K, and inference factor α on Douban (in the top row) and Yelp (in the bottom row) datasets.

G. Case Study

In our SLHRec, we could utilize the MLN to study the contribution of entities and relations related to target triplets, which enables reasoned analysis of the predictions [11]. Given a predicted triplet t, we denote $\mathcal{V}'$ and $\mathcal{E}'$ as sets of entities and relations in the Markov blanket $M(t)$, respectively. We define the set $\mathcal{T}_{e'}$ to include all triplets t' from $M(t)$ that incorporates

entity $e' \in \mathcal{V}'$. Therefore, the contribution of the entity e' on the predicted triplet t is computed as follows:

$$\text{CON}_{e'} = \sum\nolimits_{t' \in \mathcal{T}_{e'}} p\big(\mathbb{I}_{t'} | \mathbb{I}_{M(t)}\big). \tag{45}$$

Similarly, to evaluate the contribution of a specific relation r', we can replace $\mathcal{T}_{e'}$ with $\mathcal{T}_{r'}$, where set $\mathcal{T}_{r'}$ comprises all triplets in $M(t)$ that includes relation $r' \in \mathcal{E}'$. To obtain the relative contribution of each entity and relation, we could calculate their respective percentages. This can be done by dividing the individual contribution by the total sum of contributions from all entities or relations.

In Figure 9, we present case studies on Douban dataset to illustrate the reasoning about the results. First, we obtain relevant triplets by extracting the Markov blanket of the target triplet from the Markov network. The red triplets represent the observed data (assigned a probability of 1.0), while those in blue denote inferred triplets. Our aim is to predict the probability of the yellow hidden triplet and to quantify the contribution of each entity and relation to the triplet being true. In the first scenario, there are two rules f_1 and f_2 with weights of 0.98 and 0.19. Based on these rules, two cliques are established. Then, using Equation (45), we calculate the probability of the hidden triplet to be 0.223. The contributions of user_{10919}, user_{5178}, and user_{2712} to this probability are 0.270, 0.422, and 0.308, respectively. The second scenario involves the other two rules f_3 and f_4 with weights of 0.25 and 0.43. We infer that the probability of the hidden triplet is 0.378. In this case, the 'Prefer', 'Write', and 'Trust' relations contribute 0.357, 0.407, and 0.236 to this probability, respectively. Furthermore, the contributions of user_{9355}, user_{737}, author_{17396}, and book_{4667} are 0.259, 0.156, 0.343, and 0.242, respectively.

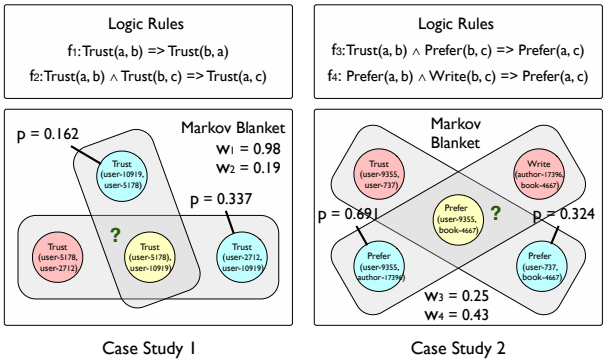

Fig. 9. Case study of reasoning by logic rules.

V. Related Work

This section introduces four relevant research.

A. HIN-based Recommendation

Heterogeneous information networks (HINs), comprising different types of nodes and links, have emerged as a robust approach for integrating complex data. Recently, HINs are increasingly applied in recommendation [35]. Generally, HIN-based recommendation is categorized into two main approaches [23]. Firstly, metapath-based methods [41], [51], [52] utilize metapaths to link users and items. For instance, NeuACF considers aspect-level latent factors from metapaths by using a deep neural network with an attention mechanism [51]. MCRec utilizes a co-attention mechanism to model the metapath-based context [21]. The second approach involves graph-based methodologies [14], [58], [60], [61], [65]. These methods leverage the concept of GNNs to learn embeddings of nodes in HINs through neighbor aggregation. However, current methods often overlook the heterophily property of HINs, which hinders their ability to capture the non-local structural information in HINs. Moreover, integrating logic rules related to HINs into their models elegantly remains a challenge. This paper introduces a HIN-based recommender SLHRec. It includes a structure-aware module which utilizes a novel dual graph neural network to consider the heterophily of HINs. Additionally, it incorporates a logic-aware module which uses MLNs to model the logic relations in HINs.

B. Heterophilic Graph Neural Networks

Most GNNs [25], [31], [57], [67] have been developed based on the homophily assumption, i.e., nodes with similar features or the same class labels are linked together. However, many real-world graphs exhibit heterophily properties, i.e., linked nodes have dissimilar features and different classes. To consider this property in graph learning, existing research methods are broadly divided into two categories [80]: non-local neighbor extension and GNN architecture refinement. The former extends local neighbors to non-local ones, primarily through high-order neighbor mixing [1] and potential neighbor discovery [30], [32], [46]. This approach enhances the representation capability of heterophilic GNNs by leveraging distant yet informative nodes. The latter category improves GNNs for heterophilic graphs by employing three strategies: adaptive message aggregation [72], ego-neighbor separation [20], and inter-layer combination [71]. These strategies aim to enhance the expressiveness of GNNs by learning distinct and discriminative node representations [80].

In this paper, our focus is on the heterophily aspect of HINs. It is worth noting that heterophily is a different concept from heterogeneity [81]. To consider this characteristic in HINs for graph learning, we adopt the approach of non-local neighbor extension [30], [32], [46]. Specifically, we employ a potential neighbor discovery method to uncover non-local structures in HINs. Additionally, we introduce a graph neural network to integrate these structural insights into node representations.

C. Statistical Relational Learning

Statistical relational learning (SRL) represents an evolving research area focused on modeling and learning in relational domains. This area contains four commonly used models [11]: probabilistic relational models (PRMs), relational dependency networks (RDNs), Bayesian logic programs (BLPs), and Markov logic networks (MLNs). Specifically, PRMs merge logical representations with probabilistic semantics, which extend Bayesian networks to accommodate relational data

[15]. RDNs is a graphical model that uses bidirected graph with conditional probability tables for variables to approximate joint distributions [43]. BLPs utilize logic programming to unify Bayesian networks, which overcomes the propositional character of Bayesian networks and logical programs [24]. MLNs combine logic rules with probabilistic graphical models [49], which manages the noise inherent in logical rules and assigns weights to represent rule confidence. Our method leverages MLNs for HIN-based recommendation. Specifically, we define the joint distribution of facts in HINs using the MLN and train it via the EM algorithm. In the further, we explore the integration of other SRL methods into our design.

D. *Contrastive Learning in Recommendation*

Contrastive learning, which is an effective strategy to learn the underlying representation and semantic information from unlabeled data [36], has been widely applied in various fields. In computer vision, SimCLR develops a contrastive loss, which aligns different perspectives of the same image to enhance visual representations [10]. In natural language processing, DeCLUTR focuses on agreement between text segments from the same source to enhance textual representations [16]. Recently, contrastive learning has received considerable attention in recommender systems [77], e.g., traditional CF-based recommendation [34], [66], sequential recommendation [68], [70], multimedia and social recommendation [13], [38], [74]. The construction of contrastive views is the key to contrastive learning-based recommender systems. One line of research leverages data augmentation to create different views. For instance, SGL generates contrastive views by node or edge dropout [66]. CL4SRec designs several data augmentation operators (i.e., cropping, masking, and reordering) for item sequences [70]. Another line of studies focuses on mining different views from the data. For instance, CrossCBR enhances bundle recommendation by using contrastive learning to align user-bundle and user-item interaction views [38]. In this paper, we leverage contrastive learning for HIN-based recommendation. Our method is closer to the second research line. Specifically, we construct two views from the structural and logical information of HINs, then design a cross-view contrastive task to distill cooperative signals between views, thereby enhancing node representation learning.

VI. Conclusion and Future Work

In this work, we introduce SLHRec, a novel recommender that leverages the graph structure and logical relations inherent in HINs. SLHRec is composed of two key components: a structure-aware module that processes the HIN at a structural level, considering HIN's heterophily nature, and a logic-aware module that models logical relations in the HIN. To enhance the cooperative association between modules, we introduce a self-supervised task. The results on four datasets shows the superior performance of our SLHRec.

In the future, we aim to advance our SLHRec in three aspects. (i) Our current method uses the non-local neighbor construction for modeling structural information. Moving forward, we plan to adopt the GNN architecture refinement strategy [80] to improve our method. (ii) In addition, while our method presently employs MLNs, we will explore the integration of other statistical relational learning models, such as PRM [15], RDNs [43], and BLP [24], into our design. (iii) Finally, to mitigate the bias amplification issues in recommender systems, we are considering the incorporation of causal inference methods, such as deconfounding [64] and counterfactual reasoning [54], into our SLHRec.

Acknowledgment

The authors thank all the reviewers for their helpful comments. This work was supported by the National Science and Technology Major Project under Grant No. 2021ZD0112500; the National Natural Science Foundation of China under Grant Nos. U22A2098, 62172185, 62206105 and 62202200.

References

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, and et al. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pages 21–29. PMLR, 2019.
[2] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications. 2021.
[3] I. Balazevic, C. Allen, and T. Hospedales. Multi-relational poincaré graph embeddings. *NIPS*, 32, 2019.
[4] Y. Bengio et al. From system 1 deep learning to system 2 deep learning. In *NIPS*, 2019.
[5] J. Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society Series D: The Statistician*, 24(3):179–195, 1975.
[6] D. Busbridge, D. Sherburn, P. Cavallo, and N. Y. Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.
[7] C. Cai and Y. Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.
[8] H. Chen, S. Shi, Y. Li, and Y. Zhang. Neural collaborative reasoning. In *WWW*, pages 1516–1527, 2021.
[9] M. Chen, C. Huang, L. Xia, and et al. Heterogeneous graph contrastive learning for recommendation. In *WSDM*, pages 544–552, 2023.
[10] T. Chen, S. Kornblith, and et al. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.
[11] Z. Chen, X. Wang, C. Wang, and J. Li. Explainable link prediction in knowledge hypergraphs. In *CIKM*, pages 262–271, 2022.
[12] P. Cui, X. Wang, J. Pei, and W. Zhu. A survey on network embedding. *TKDE*, 31(5):833–852, 2018.
[13] J. Du, Z. Ye, L. Yao, and et al. Socially-aware dual contrastive learning for cold-start recommendation. In *SIGIR*, pages 1927–1932, 2022.
[14] M. Gao, J.-Y. Li, C.-H. Chen, Y. Li, J. Zhang, and Z.-H. Zhan. Enhanced multi-task learning and knowledge graph-based recommender system. *TKDE*, 2023.
[15] L. Getoor, N. Friedman, D. Koller, and et al. Probabilistic relational models. *Introduction to statistical relational learning*, 8, 2007.
[16] J. Giorgi, O. Nitski, B. Wang, and G. Bader. Declutr: Deep contrastive learning for unsupervised textual representations.
[17] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. Deepfm: a factorization-machine based neural network for ctr prediction. 2017.
[18] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *NIPS*, 2017.
[19] L. V. Harsha Vardhan, G. Jia, and S. Kok. Probabilistic logic graph attention networks for reasoning. In *WWW*, pages 669–673, 2020.
[20] D. He, C. Liang, H. Liu, M. Wen, and et al. Block modeling-guided graph convolutional neural networks. In *AAAI*, pages 4022–4029, 2022.
[21] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *KDD*, pages 1531–1540, 2018.
[22] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *WWW*, pages 2704–2710, 2020.
[23] J. Jin, J. Qin, Y. Fang, K. Du, W. Zhang, Y. Yu, Z. Zhang, and A. J. Smola. An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In *KDD*, pages 75–84, 2020.

[24] K. Kersting and L. De Raedt. Bayesian logic programming: theory and tool. *Statistical Relational Learning*, page 291, 2007.

[25] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. 2017.

[26] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[27] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[28] A. Li, X. Liu, and B. Yang. Item attribute-aware graph collaborative filtering. *Expert Systems with Applications*, 238:122242, 2024.

[29] A. Li and B. Yang. Gsirec: Learning with graph side information for recommendation. *World Wide Web*, 24(5):1411–1437, 2021.

[30] A. Li, B. Yang, H. Huo, H. Chen, G. Xu, and Z. Wang. Hyperbolic neural collaborative recommender. *TKDE*, 2022.

[31] A. Li, B. Yang, H. Huo, and F. Hussain. Hypercomplex graph collaborative filtering. In *WWW*, pages 1914–1922, 2022.

[32] A. Li, B. Yang, H. Huo, and F. K. Hussain. Leveraging implicit relations for recommender systems. *Information Sciences*, 579:55–71, 2021.

[33] A. Li, B. Yang, F. K. Hussain, and H. Huo. Hsr: Hyperbolic social recommender. *Information Sciences*, 585:275–288, 2022.

[34] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*, pages 2320–2329, 2022.

[35] J. Liu, C. Shi, C. Yang, Z. Lu, and S. Y. Philip. A survey on heterogeneous information network based recommender systems: Concepts, methods, applications and resources. *AI Open*, 3:40–57, 2022.

[36] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive. *TKDE*, 35(1):857–876, 2021.

[37] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *KDD*, pages 1150–1160, 2021.

[38] Y. Ma, Y. He, A. Zhang, and et al. Crosscbr: cross-view contrastive learning for bundle recommendation. In *KDD*, pages 1233–1241, 2022.

[39] Q. Mao, Z. Liu, C. Liu, and J. Sun. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *WWW*, pages 599–610, 2023.

[40] G. Marcus. The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*, 2020.

[41] H. Miao, A. Li, and B. Yang. Meta-path enhanced lightweight graph neural network for social recommendation. In *DASFAA*, pages 134–149, 2022.

[42] R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[43] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8(3), 2007.

[44] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. *NIPS*, 30, 2017.

[45] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[46] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. *ICLR*, 2020.

[47] M. Qu and J. Tang. Probabilistic logic neural networks for reasoning. *NIPS*, 32, 2019.

[48] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. In *KDD*.

[49] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62:107–136, 2006.

[50] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607. Springer, 2018.

[51] C. Shi, X. Han, L. Song, X. Wang, S. Wang, J. Du, and S. Y. Philip. Deep collaborative filtering with multi-aspect information in heterogeneous networks. *TKDE*, 33(4):1413–1425, 2019.

[52] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip. Heterogeneous information network embedding for recommendation. *TKDE*, 31(2):357–370, 2018.

[53] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip. A survey of heterogeneous information network analysis. *TKDE*, 29(1):17–37, 2016.

[54] D. Shi, A. Li, and B. Yang. Counterfactual-guided and curiosity-driven multi-hop reasoning over knowledge graph. In *DASFAA*, pages 171–179, 2022.

[55] S. Shi, H. Chen, W. Ma, J. Mao, M. Zhang, and Y. Zhang. Neural logic reasoning. In *CIKM*, pages 1365–1374, 2020.

[56] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[57] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *ICLR*, 2018.

[58] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo. Knowledge graph convolutional networks for recommender systems. In *WWW*.

[59] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and S. Y. Philip. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 9(2):415–436, 2022.

[60] X. Wang, X. He, Y. Cao, and et al. Kgat: Knowledge graph attention network for recommendation. In *KDD*, pages 950–958, 2019.

[61] X. Wang, T. Huang, D. Wang, Y. Yuan, Z. Liu, X. He, and T.-S. Chua. Learning intents behind interactions with knowledge graph for recommendation. In *WWW*, pages 878–887, 2021.

[62] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *WWW*, pages 2022–2032, 2019.

[63] X. Wang, N. Liu, H. Han, and C. Shi. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *KDD*.

[64] Y. Wang, D. Liang, L. Charlin, and D. M. Blei. The deconfounded recommender: A causal inference approach to recommendation. *arXiv preprint arXiv:1808.06581*, 2018.

[65] Z. Wang, G. Lin, H. Tan, Q. Chen, and X. Liu. Ckan: Collaborative knowledge-aware attentive network for recommender systems. In *SIGIR*, pages 219–228, 2020.

[66] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie. Self-supervised graph learning for recommendation. In *SIGIR*, pages 726–735, 2021.

[67] R. Xia, C. Zhang, A. Li, X. Liu, and B. Yang. Attribute imputation autoencoders for attribute-missing graphs. *KBS*, page 111583, 2024.

[68] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang. Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI*, pages 4503–4511, 2021.

[69] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. 2017.

[70] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, and B. Cui. Contrastive learning for sequential recommendation. In *ICDE*, pages 1259–1273, 2022.

[71] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5453–5462. PMLR, 2018.

[72] L. Yang, M. Li, L. Liu, C. Wang, and et al. Diverse message passing for attribute with heterophily. *NIPS*, 34:4751–4763, 2021.

[73] X. Yang, M. Yan, S. Pan, and et al. Simple and efficient heterogeneous graph neural network. In *AAAI*, pages 10816–10824, 2023.

[74] Y. Yang, C. Huang, L. Xia, and C. Li. Knowledge graph contrastive learning for recommendation. 2022.

[75] D. Yu, X. Liu, S. Pan, A. Li, and B. Yang. A novel neural-symbolic system under statistical relational learning. *arXiv preprint arXiv:2309.08931*, 2023.

[76] D. Yu, B. Yang, Q. Wei, A. Li, and S. Pan. A probabilistic graphical model based on neural-symbolic reasoning for visual relationship detection. In *CVPR*, pages 10609–10618, 2022.

[77] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang. Self-supervised learning for recommender systems: A survey. *TKDE*, 2023.

[78] X. Zang, B. Yang, X. Liu, and A. Li. Dnea: Dynamic network embedding method for anomaly detection. In *KSEM*, pages 236–248, 2021.

[79] Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, and L. Song. Efficient probabilistic logic reasoning with graph neural networks. In *ICLR*, 2020.

[80] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[81] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *NIPS*, pages 7793–7804, 2020.