Self-Supervised Hypergraph Transformer for Recommender Systems

Lianghao Xia
University of Hong Kong
Hong Kong, China
aka_xia@foxmail.com

Chao Huang*
University of Hong Kong
Hong Kong, China
chaohuang75@gmail.com

Chuxu Zhang
Brandeis University
Waltham, USA
chuxuzhang@brandeis.edu

ABSTRACT

Graph Neural Networks (GNNs) have been shown as promising solutions for collaborative filtering (CF) with the modeling of user-item interaction graphs. The key idea of existing GNN-based recommender systems is to recursively perform the message passing along the user-item interaction edge for refining the encoded embeddings. Despite their effectiveness, however, most of the current recommendation models rely on sufficient and high-quality training data, such that the learned representations can well capture accurate user preference. User behavior data in many practical recommendation scenarios is often noisy and exhibits skewed distribution, which may result in suboptimal representation performance in GNN-based models. In this paper, we propose SHT, a novel Self-Supervised Hypergraph Transformer framework (SHT) which augments user representations by exploring the global collaborative relationships in an explicit way. Specifically, we first empower the graph neural CF paradigm to maintain global collaborative effects among users and items with a hypergraph transformer network. With the distilled global context, a cross-view generative self-supervised learning component is proposed for data augmentation over the user-item interaction graph, so as to enhance the robustness of recommender systems. Extensive experiments demonstrate that SHT can significantly improve the performance over various state-of-the-art baselines. Further ablation studies show the superior representation ability of our SHT recommendation framework in alleviating the data sparsity and noise issues. The source code and evaluation datasets are available at: https://github.com/akaxlh/SHT.

CCS CONCEPTS

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

KEYWORDS

Self-Supervised Learning, Graph Neural Networks, Hypergraph Representation, Recommender System

*Chao Huang is the corresponding author.

KDD'22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539473

ACM Reference Format:
Lianghao Xia, Chao Huang, and Chuxu Zhang. 2022. Self-Supervised Hypergraph Transformer for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'22), August 14–18, 2022, Washington, DC, USA.* ACM, Washington DC, 10 pages. https://doi.org/10.1145/3534678.3539473

1 INTRODUCTION

Recommender systems have become increasingly important to alleviate the information overload for users in a variety of web applications, such as e-commerce systems [5], streaming video sites [17] and location-based lifestyle apps [4]. To accurately infer the user preference, encoding user and item informative representations is the core part of effective collaborative filtering (CF) paradigms based on the observed user-item interactions [7, 8, 20].

Earlier CF models project interaction data into latent user and item embeddings using matrix factorization (MF) [13]. Due to the strong representation ability of deep learning, various neural network CF models have been developed to project users and items into latent low-dimensional representations, such as autoencoder [15] and attention mechanism [2]. Recent years have witnessed the development of graph neural networks (GNNs) for modeling graph-structural data [27, 30]. One promising direction is to perform the information propagation along the user-item interactions to refine user embeddings based on the recursive aggregation schema. For example, upon the graph convolutional network, PinSage [38] and NGCF [26] attempt to aggregate neighboring information by capturing the graph-based CF signals for recommendation. To simplify the graph-based message passing, LightGCN [6] omits the burdensome non-linear transformer during the embedding propagation and improve the recommendation performance. To further enhance the graph-based user-item interaction modeling, some follow-up studies propose to learn intent-aware representations with disentangled graph neural frameworks (*e.g.*, DisenHAN [29]), differentiate behavior-aware embeddings of users with multi-relational graph neural models (*e.g.*, MB-GMN [34]).

Despite the effectiveness of the above graph-based CF models by providing state-of-the-art recommendation performance, several key challenges have not been well addressed in existing methods. *First*, data noise is ubiquitous in many recommendation scenarios due to a variety of factors. For example, users may click their uninterested products due to the over-recommendation of popular items [42]. In such cases, the user-item interaction graph may contain " interest-irrelevant" connections. Directly aggregating information from all interaction edges will impair the accurate user representation. Worse still, the embedding propagation among multi-hop adjacent vertices (user or item) will amplify the noise

effects, which misleads the encoding of underlying user interest in GNN-based recommender systems. *Second*, data sparsity and skewed distribution issue still stand in the way of effective user-item interaction modeling, leading to most existing graph-based CF models being biased towards popular items [14, 41]. Hence, the recommendation performance of current approaches severely drops with the user data scarcity problem, as the high-quality training signals could be small. While there exist a handful of recently developed recommendation methods (SGL [31] and SLRec [37]) leveraging self-supervised learning to improve user representations, these methods mainly generate the additional supervision information with probability-based randomly mask operations, which might keep some noisy interaction and dropout some important training signals during the data augmentation process.

Contribution. In light of the aforementioned challenges, this work proposes a Self-Supervised Hypergraph Transformer (SHT) to enhance the robustness and generalization performance of graph-based CF paradigms for recommendation. Specifically, we integrate the hypergraph neural network with the topology-aware Transformer, to empower our SHT to maintain the global cross-user collaborative relations. Upon the local graph convolutional network, we first encode the topology-aware user embeddings and inject them into Transformer architecture for hypergraph-guided message passing within the entire user/item representation space.

In addition, we unify the modeling of local collaborative relation encoder with the global hypergraph dependency learning under a generative self-supervised learning framework. Our proposed new self-supervised recommender system distills the auxiliary supervision signals for data augmentation through a graph topological denoising scheme. A graph-based meta transformation layer is introduced to project hyergraph-based global-level representations into the graph-based local-level interaction modeling for user and item dimensions. Our new proposed SHT is a model-agnostic method and serve as a plug-in learning component in existing graph-based recommender systems. Specifically, SHT enables the cooperation of the local-level and global-level collaborative relations, to facilitate the graph-based CF models to learn high-quality user embeddings from noisy and sparse user interaction data.

The key contributions of this work are summarized as follows:

- In this work, we propose a new self-supervised recommendation model–SHT to enhance the robustness of graph collaborative filtering paradigms, by integrating the hypergraph neural network with the topology-aware Transformer.
- In the proposed SHT method, the designed hypergraph learning component encodes the global collaborative effects within the entire user representation space, via a learnable multi-channel hyperedge-guided message passing schema. Furthermore, the local and global learning views for collaborative relations are integrated with the cooperative supervision for interaction graph topological denoising and auxiliary knowledge distillation.
- Extensive experiments demonstrate that our proposed SHT framework achieves significant performance improvement over 15 different types of recommendation baselines. Additionally, we conduct empirical analysis to show the rationality of our model design with the ablation studies.

2 PRELIMINARIES AND RELATED WORK

Recap Graph Collaborative Filtering Paradigm. To enhance the Collaborative Filtering with the multi-order connectivity information, one prominent line of recommender systems generates graph structures for user-item interactions. Suppose our recommendation scenario involves I users and J items with the user set $\mathcal{U} = \{u_1, ...u_I\}$ and item set $\mathcal{V} = \{v_1, ...v_J\}$. Edges in the user-item interaction graph $\mathcal{G}$ are constructed if user u_i has adopted item v_j. Upon the constructed interaction graph structures, the core component of graph-based CF paradigm lies in the information aggregation function–gathering the feature embeddings of neighboring users/items via different aggregators, *e.g.*, mean or sum.

Recommendation with Graph Neural Networks. Recent studies have attempted to design various graph neural architectures to model the user-item interaction graphs through embedding propagation. For example, PinSage [38] and NGCF [26] are built upon the graph convolutional network over the spectral domain. Later on, LightGCN [6] proposes to simplify the heavy non-linear transformation and utilizes the sum-based pooling over neighboring representations. Upon the GCN-based message passing schema, each user and item is encoded into the transformed embeddings with the preservation of multi-hop connections. To further improve the user representation, some recent studies attempt to design disentangled graph neural architecture for user-item interaction modeling, such as DGCF [28] and DisenHAN [29]. Several multi-relational GNNs are proposed to enhance recommender systems with multi-behavior modeling, including KHGT [32] and HMG-CR [35]. However, most of existing graph neural CF models are intrinsic designed to merely rely on the observation interaction lables for model training, which makes them incapable of effectively modeling interaction graph with sparse and noisy supervision signals. To overcome these challenges, this work proposes a self-supervised hypergraph transformer architecture to generate informative knowledge through the effective interaction between local and global collaborative views.

Hypergraph-based Recommender Systems. There exist some recently developed models constructing hypergraph connections to improve the relation learning for recommendation [11, 25, 39]. For example, HyRec [25] regards users as hyperedges to aggregate information from the interacted items. MHCN [39] constructs multi-channel hypergraphs to model high-order relationships among users. Furthermore, DHCF [11] is a hypergraph collaborative filtering model to learn the hybrid high-order correlations. Different from these work for generating hypergraph structures with manually design, this work automates the hypergraph structure learning process with the modeling of global collaborative relation.

Self-Supervised Graph Learning. To improve the embedding quality of supervised learning, self-supervised learning (SSL) has become a promising solution with auxiliary training signals [16], such as augmented image data [12], pretext sequence tasks for language data [24], knowledge graph augmentation [36]. Recently, self-supervised learning has also attracted much attention on graph representation [10]. For example, DGI [23] and GMI [18] perform the generative self-supervised learning over the GNN framework with auxiliary tasks. Inspired by the graph self-supervised learning,

SGL [31] produces state-of-the-art performance by generating contrastive views with randomly node and edge dropout operations. Following this research line, HCCF [33] leverages the hypergraph to generate contrastive signals to improve the graph-based recommender system. Different from them, this work enhances the graph-based collaborative filtering paradigm with a generative self-supervised learning framework.

3 METHODOLOGY

In this section, we present the proposed SHT framework and show the overall model architecture in Figure 1. SHT embeds local structure information into latent node representations, and conduct global relation learning with the local-aware hypergraph transformer. To train the proposed model, we augment the regular parameter learning with the local-global cross-view self-augmentation.

3.1 Local Graph Structure Learning

To begin with, we embed users and items into a d-dimensional latent space to encode their interaction patterns. For user u_i and item v_j, embedding vectors $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^d$ are generated, respectively. Also, we aggregate all the user and item embeddings to compose embedding matrices $\mathbf{E}^{(u)} \in \mathbb{R}^{I\times d}, \mathbf{E}^{(v)} \in \mathbb{R}^{J\times d}$, respectively. We may omit the superscript (u) and (v) for notation simplification when it is not important to differentiate the user and item index.

Inspired by recent success of graph convolutional networks [6, 30] in capturing local graph structures, we propose to encode the neighboring sub-graph structure of each node into a graph topology-aware embedding, to inject the topology positional information into our graph transformer. Specifically, SHT employs a two-layer light-weight graph convolutional network as follows:

$$\bar{\mathbf{E}}^{(u)} = \mathrm{GCN}^2(\mathbf{E}^{(v)}, \mathcal{G}) = \bar{\mathcal{A}} \cdot \bar{\mathcal{A}}^\top \mathbf{E}^{(u)} + \bar{\mathcal{A}} \cdot \mathbf{E}^{(v)} \tag{1}$$

where $\bar{\mathbf{E}}^{(u)} \in \mathbb{R}^{I\times d}$ denotes the topology-aware embeddings for users. $\mathrm{GCN}^2(\cdot)$ denotes two layers of message passing. $\bar{\mathcal{A}} \in \mathbb{R}^{I\times J}$ refers to the normalized adjacent matrix of graph $\mathcal{G}$, which is calculated by $\bar{\mathcal{A}}_{i,j} = \mathcal{A}_{i,j}/(\mathbf{D}_i^{(u)1/2}\mathbf{D}_j^{(v)1/2})$, where $\mathcal{A}$ is the original binary adjacent matrix. $\mathbf{D}_i^{(u)}, \mathbf{D}_j^{(v)}$ refer to the degree of u_i and v_j in graph $\mathcal{G}$, respectively. Note that SHT considers neighboring nodes in different distance through residual connections. The topology-aware embeddings for items can be calculated analogously.

3.2 Hypergraph Transformer for Global Relation Learning

Though existing graph-based neural networks have shown their strength in learning interaction data [3, 6, 26], the inherent noise and skewed data distribution in recommendation scenario limit the performance of graph representation for user embeddings. To address this limitation, SHT adopts a hypergraph transformer framework, which i) alleviates the noise issue by enhancing the user collaborative relation modeling with the adaptive hypergraph relation learning; ii) transfer knowledge from dense user/item nodes to sparse ones. Concretely, SHT is configured with a Transformer-like attention mechanism for structure learning. The encoded graph topology-aware embeddings are injected into the node representations to preserve the graph locality and topological positions. Meanwhile, the multi-channel attention [22] further benefits our structure learning in SHT.

In particular, SHT generates input embedding vectors for u_i and v_j by combining the id-corresponding embeddings $(\mathbf{e}_i, \mathbf{e}_j)$ together with the topology-aware embeddings (vectors $\bar{\mathbf{e}}_i, \bar{\mathbf{e}}_j$ from embedding tables $\bar{\mathbf{E}}^{(u)}$ and $\bar{\mathbf{E}}^{(v)}$) as follows:

$$\tilde{\mathbf{e}}_i = \mathbf{e}_i + \bar{\mathbf{e}}_i; \quad \tilde{\mathbf{e}}_j = \mathbf{e}_j + \bar{\mathbf{e}}_j \tag{2}$$

Then, SHT conducts hypergraph-based information propagation as well as hypergraph structure learning using $\tilde{\mathbf{e}}_i, \tilde{\mathbf{e}}_j$ as input. We utilize K hyperedges to distill the collaborative relations from the global perspective. Node embeddings are propagated to each other using hyperedges as intermediate hubs, where the connections between nodes and hyperedges are optimized to reflect the implicit dependencies among nodes.

3.2.1 **Node-to-Hyperedge Propagation.** Without loss of generality, we mainly discuss the information propagation between user nodes and user-side hyperedges for simplicity. The same process is applied for item nodes analogously. The propagation from user nodes to user-side hyperedges can be formally presented as follows:

$$\tilde{\mathbf{z}}_k = \Big\Vert_{h=1}^{H} \bar{\mathbf{z}}_{k,h}; \quad \bar{\mathbf{z}}_{k,h} = \sum_{i=1}^{I} \mathbf{v}_{i,h}\mathbf{k}_{i,h}^\top \mathbf{q}_{k,h} \tag{3}$$

where $\tilde{\mathbf{z}}_k \in \mathbf{R}^d$ denotes the embedding for the k-th hyperedge. It is calculated by concatenating the H head-specific hyperedge embeddings $\bar{\mathbf{z}}_{k,h} \in \mathbb{R}^{d/H}$. $\mathbf{q}_{k,h}, \mathbf{k}_{i,h}, \mathbf{v}_{i,h} \in \mathbb{R}^{d/H}$ are the query, key and value vectors in the attention mechanism which will be elaborated later. Here, we calculate the edge weight between hyperedge k and user u_i through a linear dot-product $\mathbf{k}_{i,h}^\top \mathbf{q}_{k,h}$, which reduces the complexity from $O(K\times I\times d/H)$ to $O((I+K)\times d^2/H^2)$ by avoiding directly calculating the node-hyperedge connections (*i.e.* $\mathbf{k}_{i,h}^\top \mathbf{q}_{k,h}$), but the key-value dot-product first (*i.e.* $\sum_{i=1}^{I} \mathbf{v}_{i,h}\mathbf{k}_{i,h}^\top$).

In details, the multi-head query, key and value vectors are calculated through linear transformations and slicing. The h-head-specific embeddings are calculated by:

$$\mathbf{q}_{k,h} = \mathbf{Z}_{k,p_{h-1}:p_h}; \quad \mathbf{k}_{i,h} = \mathbf{K}_{p_{h-1}:p_h,:}\tilde{\mathbf{e}}_i; \quad \mathbf{v}_{i,h} = \mathbf{V}_{p_{h-1}:p_h,:}\tilde{\mathbf{e}}_i \tag{4}$$

where $\mathbf{q}_{k,h} \in \mathbb{R}^{d/H}$ denotes the h-head-specific query embedding for the k-th hyperedge, $\mathbf{k}_{i,h}, \mathbf{v}_{i,h} \in \mathbb{R}^{d/H}$ denotes the h-head-specific key and value embedding for user u_i. $\mathbf{Z} \in \mathbb{R}^{K\times d}$ represents the embedding matrix for the K hyperedges. $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{d\times d}$ represents the key and the value transformation of all the H heads, respectively. $p_{h-1} = \frac{(h-1)d}{H}$ and $p_h = \frac{hd}{H}$ denote the start and the end indices of the h-th slice.

To further excavate the complex non-linear feature interactions among the hyperedges, SHT is augmented with two-layer hierarchical hypergraph neural networks for both user side and item side. Specifically, the final hyperedge embeddings are calculated by:

$$\hat{\mathbf{Z}} = \mathrm{HHGN}^2(\tilde{\mathbf{Z}}); \quad \mathrm{HHGN}(\mathbf{X}) = \sigma(\mathcal{H} \cdot \mathbf{X} + \mathbf{X}) \tag{5}$$

where $\hat{\mathbf{Z}}, \tilde{\mathbf{Z}} \in \mathbb{R}^{K\times d}$ represent the embedding tables for the final and the original hyperedge embeddings, consisting of hyperedge-specific embedding vectors $\hat{\mathbf{z}}, \tilde{\mathbf{z}} \in \mathbb{R}^d$, respectively. $\mathrm{HHGN}^2(\cdot)$ denotes applying the hierarchical hypergraph network (HHGN) twice.

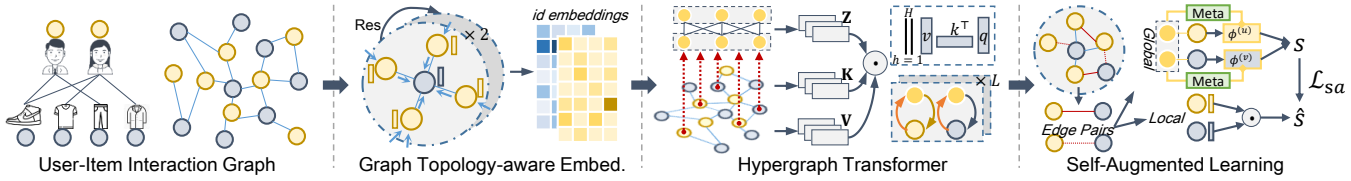


Figure 1: Overall framework of the proposed SHT model.

HHGN is configured with a learnable parametric matrix $\mathcal{H} \in \mathbb{R}^{K \times K}$, which characterizes the hyperedge-wise relations. An activation function $\sigma(\cdot)$ is introduced for non-linear relation modeling. Additionally, we utilize a residual connection to facilitate gradient propagation in our hypergraph neural structures.

3.2.2 **Hyperedge-to-Node Propagation.** With the final hyperedge embeddings $\hat{\mathbf{Z}}$, we propagate the information from hyperedges to user/item nodes through a similar but reverse process:

$$\tilde{\mathbf{e}}'_i = \Big\Vert_{h=1}^{H} \bar{\mathbf{e}}'_{i,h}; \quad \bar{\mathbf{e}}'_{i,h} = \sum_{k=1}^{K} \mathbf{v}'_{k,h} \mathbf{k}'^{\top}_{k,h} \mathbf{q}'_{i,h} \tag{6}$$

where $\tilde{\mathbf{e}}'_i \in \mathbb{R}^d$ denotes the new embedding for user u_i refined by the hypergraph neural network. $\bar{\mathbf{e}}'_{i,h} \in \mathbb{R}^{d/H}$ denotes the node embedding calculated by the h-th attention head for u_i. $\mathbf{q}'_{i,h}, \mathbf{k}'_{k,h}, \mathbf{v}'_{k,h} \in \mathbb{R}^{d/H}$ represent the query, key and value vectors for user u_i and hyperedge k. The attention calculation in this hyperedge-to-node propagation process shares most parameters with the aforementioned node-to-hyperedge propagation. The former query serves as key, and the former key serves as query here. The value calculation applies the same value transformation for the hyperedge embedding. The calculation process can be formally stated as:

$$\mathbf{q}'_{i,h} = \mathbf{k}_{i,h}; \quad \mathbf{k}'_{k,h} = \mathbf{q}_{k,h}; \quad \mathbf{v}'_{k,h} = \mathbf{V}_{p_{h-1}:p_h,:} \hat{\mathbf{z}}_k \tag{7}$$

3.2.3 **Iterative Hypergraph Propagation.** Based on the prominent node-wise relations captured by the learned hypergraph structures, we propose to further propagate the encoded global collaborative relations via stacking multiple hypergraph transformer layers. In this way, the long-range user/item dependencies can be characterized by our SHT framework through the iterative hypergraph propagation. In form, taking the embedding tables $\tilde{\mathbf{E}}_{l-1}$ in the $(l-1)$-th iteration as input, SHT recursively applies the hypergraph encoding (denoted by HyperTrans$(\cdot)$) and obtains the final node embeddings $\hat{\mathbf{E}} \in \mathbb{R}^{I \times d}$ or $\mathbb{R}^{J \times d}$ as follows:

$$\tilde{\mathbf{E}}_l = \text{HyperTrans}(\tilde{\mathbf{E}}_{l-1}); \quad \hat{\mathbf{E}} = \sum_{l=1}^{L} \tilde{\mathbf{E}}_l \tag{8}$$

where the layer-specific embeddings are combined through element-wise summation. The iterative hypergraph propagation is identical for the user nodes and item nodes. Finally, SHT makes predictions through dot product as $p_{i,j} = \hat{\mathbf{e}}_i^{(u)\top} \hat{\mathbf{e}}_j^{(v)}$, where $p_{i,j}$ is the forecasting score denoting the probability of u_i interacting with v_j.

3.3 Local-Global Self-Augmented Learning

The foregoing hypergraph transformer addresses the data sparsity problem through adaptive hypergraph message passing. However, the graph topology-aware embedding for local collaborative relation modeling may still be affected by the interaction data noise. To tackle this challenge, we propose to enhance the model training with self-augmented learning between the local topology-aware embedding and the global hypergraph learning. To be specific, the topology-aware embedding for local information extraction is augmented with an additional task to differentiate the solidity of sampled edges in the observed user-item interaction graph. Here, solidity refers to the probability of an edge not being noisy, and its label in the augmented task is calculated based on the learned hypergraph dependencies and representations. In this way, SHT transfers knowledge from the high-level and denoised features in the hypergraph transformer, to the low-level and noisy topology-aware embeddings, which is expected to recalibrate the local graph structure and improve the model robustness. The workflow of our self-augmented module is illustrated in Fig 2.

3.3.1 **Solidity Labeling with Meta Networks.** In our SHT model, the learned hypergraph dependency representations can serve as useful knowledge to denoise the observed user-item interactions by associating each edge with a learned solidity score. Specifically, we reuse the key embeddings $\mathbf{k}_{i,h}, \mathbf{k}_{j,h}$ in Eq 4 to represent user u_i and item v_j when estimating the solidity score for the edge (u_i, v_j). This is because that the key vectors are generated for relation modeling and can be considered as helpful information source for interaction solidity estimation. Furthermore, we propose to also take the hyperedge embeddings $\mathbf{Z} \in \mathbb{R}^{K \times d}$ in Eq 4 into consideration, to introduce global characteristics into the solidity labeling.

Concretely, we first concatenate the multi-head key vectors and apply a simple perceptron to eliminate the gap between user/item-hyperedge relation learning and user-item relation learning. Formally, the updated user/item embeddings are calculated by:

$$\mathbf{\Gamma}_i = \phi^{(u)}\left(\Big\Vert_{h=1}^{H} \mathbf{k}_{i,h}\right); \quad \mathbf{\Gamma}_j = \phi^{(v)}\left(\Big\Vert_{h=1}^{H} \mathbf{k}_{j,h}\right) \tag{9}$$

where $\phi^{(u)}(\cdot), \phi^{(v)}(\cdot)$ are the user- and item-specific perceptrons for feature vector transformation, respectively. This projection is conducted with a meta network, using the user-side and the item-side hyperedge embeddings as input individually:

$$\phi(\mathbf{x}; \mathbf{Z}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}); \quad \mathbf{W} = \mathbf{V}_1 \bar{\mathbf{z}} + \mathbf{W}_0; \quad \mathbf{b} = \mathbf{V}_2 \bar{\mathbf{z}} + \mathbf{b}_0 \tag{10}$$

where $\mathbf{x} \in \mathbb{R}^d$ denotes the input user/item key embedding (*e.g.*, $\mathbf{\Gamma}_i, \mathbf{\Gamma}_j$). $\phi(\cdot)$ being user-specific or item-specific depends on $\mathbf{Z}$ being user-side or item-side hyperedge embedding table. $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are the parameters generated by the meta network according to the input $\mathbf{Z}$. In this way, the parameters are generated based on the learned hyperedge embeddings, which encodes global features of user- or item-specific hypergraphs. $\bar{\mathbf{z}} \in \mathbb{R}^d$ denotes

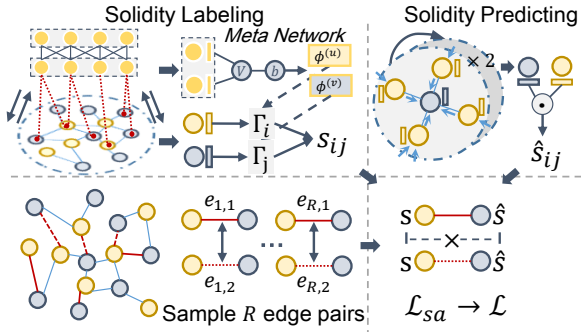


Figure 2: Workflow of the self-augmented learning.

the mean pooling of hyperedge embeddings (*i.e.* $\bar{\mathbf{z}} = \sum_{k=1}^{K} \mathbf{z}_k/K$). $\mathbf{V}_1 \in \mathbb{R}^{d\times d\times d}, \mathbf{W}_0 \in \mathbb{R}^{d\times d}, \mathbf{V}_2 \in \mathbb{R}^{d\times d}, \mathbf{b}_0 \in \mathbb{R}^d$ are the parameters of the meta network.

With the updated user/item embeddings $\mathbf{\Gamma}_i, \mathbf{\Gamma}_j$, SHT then calculates the solidity labels for edge (u_i, v_j) through a two-layer neural network as follows:

$$s_{i,j} = \text{sigm}(\mathbf{d}^\top \cdot \sigma(\mathbf{T} \cdot [\mathbf{\Gamma}_i; \mathbf{\Gamma}_j] + \mathbf{\Gamma}_i + \mathbf{\Gamma}_j + \mathbf{c})) \tag{11}$$

where $s_{i,j} \in \mathbb{R}$ denotes the solidity score given by the hypergraph transformer. $\text{sigm}(\cdot)$ denotes the sigmoid function which limits the value range of $s_{i,j}$. $\mathbf{d} \in \mathbb{R}^d, \mathbf{T} \in \mathbb{R}^{d\times 2d}, \mathbf{c} \in \mathbb{R}^d$ are the parametric matrices or vectors. $[\cdot;\cdot]$ denotes the vector concatenation.

3.3.2 **Pair-wise Solidity Ranking.** To enhance the optimization of topological embeddings, SHT employs an additional objective function to better estimate the edge solidity using the above $s_{i,j}$ as training labels. In particular, R pairs of edges $\{(e_{1,1}, e_{1,2}),...,(e_{R,1}, e_{R,2})\}$ from the observed edges in $\mathcal{G}$ are sampled, and SHT gives predictions on the solidity using the topology-aware embeddings. The predictions are then updated by optimizing the loss below:

$$\mathcal{L}_{sa} = \sum_{r=1}^{R} \max(0, 1 - (\hat{s}_{u_{r,1},v_{r,1}} - \hat{s}_{u_{r,2},v_{r,2}})(s_{u_{r,1},v_{r,1}} - s_{u_{r,2},v_{r,2}}));$$
$$\hat{s}_{u_{r,1},v_{r,1}} = \mathbf{e}_{u_{r,1}}^\top \mathbf{e}_{v_{r,1}}; \quad \hat{s}_{u_{r,2},v_{r,2}} = \mathbf{e}_{u_{r,2}}^\top \mathbf{e}_{v_{r,2}} \tag{12}$$

where $\mathcal{L}_{sa}$ denotes the loss function for our self-augmented learning. $\hat{s}_{u_{r,1},v_{r,1}}, \hat{s}_{u_{r,2},v_{r,2}}$ denote the solidity scores predicted by the topology-aware embedding, while $s_{u_{r,1},v_{r,1}}, s_{u_{r,2},v_{r,2}}$ denote the edge solidity labels given by the hypergraph transformer. Here $u_{r,1}$ and $v_{r,1}$ represent the user and the item node of edge $e_{r,1}$, respectively.

In the above loss function, the label term $(s_{u_{r,1},v_{r,1}} - s_{u_{r,2},v_{r,2}})$ not only indicates the sign of the difference (*i.e.* which one of $e_{r,1}$ and $e_{r,2}$ is bigger), but also indicates how bigger the difference is. In this way, if the solidity labels for a pair of edges given by the hypergraph transformer are close to each other, then the gradients on the predicted solidity scores given by the topology-aware embedding will become smaller. In this way, SHT is self-augmented with an adaptive ranking task, to further refine the low-level topology-aware embeddings using the high-level embeddings encoded from the hypergraph transformer.

3.4 Model Learning

We train our SHT by optimizing the main task on implicit feedback together with the self-augmented ranking task. Specifically, R' positive edges (observed in $\mathcal{G}$) and R' negative edges (not observed in $\mathcal{G}$) are sampled $\{(e_{1,1}, e_{1,2}), (e_{2,1}, e_{2,2})..., (e_{R',1}, e_{R',2})\}$, where $e_{r,1}$ and $e_{r,2}$ are individual positive and negative sample, respectively. The following pair-wise marginal objective function is applied:

$$\mathcal{L} = \sum_{r=1}^{R'} \max(0, 1 - (p_{u_{r,1},v_{r,1}} - p_{u_{r,2},v_{r,2}})) + \lambda_1 \mathcal{L}_{\text{sa}} + \lambda_2 \|\mathbf{\Theta}\|_{\text{F}}^2 \tag{13}$$

where $p_{u_{r,1},v_{r,1}}$ and $p_{u_{r,2},v_{r,2}}$ are prediction scores for edge $e_{r,1}$ and $e_{r,2}$, respectively. λ_1 and λ_2 are weights for different loss terms. $\|\mathbf{\Theta}\|_{\text{F}}^2$ denotes the l_2 regularization term for weight decay.

3.4.1 **Complexity Analysis.** We compare our SHT framework with several state-of-the-art approaches on collaborative filtering, including graph neural architectures (*e.g.* NGCF [26], LightGCN [6]) and hypergraph neural networks(*e.g.* DHCF [11]). As discussed before, our hypergraph transformer enables the complexity reduction from $O(K \times (I + J) \times d)$ to $O((I + J + K) \times d^2)$. As the typical value of the number of hyperedge K is much smaller than the number of nodes I and J, but larger than the embedding size d, the latter term is smaller and close to $O((I+J)\times d^2)$. In comparison, the complexity for a typical graph neural architecture is $O(M \times d + (I + J) \times d^2)$. So our hypergraph transformer network can achieve comparable efficiency as GNNs, such as graph convolutional networks in model inference. The existing hypergraph-based methods commonly preprocess high-order node relations to construct hypergraphs, which makes them usually more complex than the graph neural networks. In our SHT, the self-augmented task with the loss $\mathcal{L}_{\text{sa}}$ has the same complexity with the original main task.

Table 1: Statistical information of the experimental datasets.

Stat.	Yelp	Gowalla	Tmall
# Users	29601	50821	47939
# Items	24734	24734	41390
# Interactions	1517326	1069128	2357450
Density	2.1×10^{-3}	4.0×10^{-4}	1.2×10^{-3}

4 EVALUATION

To evaluate the effectiveness of our SHT, our experiments are designed to answer the following research questions:

- **RQ1**: How does our SHT perform by comparing to strong baseline methods of different categories under different settings?
- **RQ2**: How do the key components of SHT (*e.g.*, the hypergraph modeling, the transformer-like information propagation) contribute to the overall performance of SHT on different datasets?
- **RQ3**: How well can our SHT handle noisy and sparse data, as compared to baseline methods?
- **RQ4**: In real cases, can the designed the self-supervised learning mechanism in SHT provide useful interpretations?

4.1 Experimental Settings

4.1.1 **Experimental Datasets.** The experiments are conducted on three datasets collected from real-world applications, *i.e.*, Yelp, Gowalla and Tmall. The statistics of them are shown in Table 1.

- **Yelp**: This commonly-used dataset contains user ratings on business venues collected from Yelp. Following other papers on implicit feedback [9], we treat users' rated venues as interacted items and treat unrated venues as non-interacted items.

- **Gowalla**: It contains users' check-in records on geographical locations obtained from Gowalla. This evaluation dataset is generated from the period between 2016 and 2019.
- **Tmall**: This E-commerce dataset is released by Tmall, containing users' behaviors for online shopping. We collect the page-view interactions during December in 2017.

4.1.2 **Evaluation Protocols.** Following the recent collaborative filtering models [6, 31], we split the datasets by 7:2:1 into training, validation and testing sets. We adopt all-rank evaluation protocol. When testing a user, the positive items in the test set and all the non-interacted items are tested and ranked together. We employ the commonly-used *Recall@N* and *Normalized Discounted Culmulative Gain (NDCG)@N* as evaluation metrics for recommendation performance evaluation [19, 26]. N is set as 20 by default.

4.1.3 **Compared Baseline Methods.** We evaluate our SHT by comparing it with 15 baselines from different research lines for comprehensive evaluation.

Traditional Factorization-based Technique.

- **BiasMF** [13]: This method augments matrix factorization with user and item bias vectors to enhance user-specific preferences.

Neural Factorization Method.

- **NCF** [7]: This method replaces the dot-product in conventional matrix factorization with multi-layer neural networks. Here, we adopt the NeuMF variant for comparison.

Autoencoder-based Collaborative Filtering Approach.

- **AutoR** [21]: It improves user/item representations with a three-layer autoencoder trained under a behavior reconstruction task.

Graph Neural Networks for Recommendation.

- **GCMC** [1]: This is one of the pioneering work to apply graph convolutional networks (GCNs) to the matrix completion task.
- **PinSage** [38]: It applies random sampling in graph convolutional framework to study the collaborative filtering task .
- **NGCF** [26]: This graph convolution-based approach additionally takes source-target representation interaction learning into consideration when designing its graph encoder.
- **STGCN** [40]: The model combines conventional graph convolutional encoders with graph autoencoders to improve the model robustness against sparse and cold-start samples.
- **LightGCN** [6]: This work conducts in-depth analysis to study the effectiveness of modules in standard GCN for collaborative data, and proposes a simplified GCN model for recommendation.
- **GCCF** [3]: This is another method which simplifies the GCNs by removing the non-linear transformation. In GCCF, the effectiveness of residual connections across graph iterations is validated.

Disentangled Graph Model for Recommendation.

- **DGCF** [28]: It disentangles user interactions into multiple latent intentions to model user preference in a fine-grained way.

Hypergraph-based Neural Collaborative Filtering.

- **HyRec** [25]: This is a sequential collaborative model that learns item-wise high-order relations with hypergraphs.
- **DHCF** [11]: This model adopts dual-channel hypergraph neural networks for both users and items in collaborative filtering.

Recommenders enhanced by Self-Supervised Learning .

- **MHCN** [39]: This model maximizes the mutual information between node embeddings and global readout representations, to regularize the representation learning for interaction graph.
- **SLRec** [37]: This approach employs the contrastive learning between the node features as regularization terms to enhance the existing recommender systems.
- **SGL** [31]: This model conducts data augmentation through random walk and feature dropout to generate multiple views. It enhances LightGCN with self-supervised contrastive learning.

4.1.4 **Implementation Details.** We implement our SHT using TensorFlow and use Adam as the optimizer for model training with the learning rate of $1e^{-3}$ and 0.96 epoch decay ratio. The models are configured with 32 embedding dimension size, and the number of graph neural layers is searched from {1,2,3}. The weights λ_1, λ_2 for regularization terms are selected from $\{a \times 10^{-x} : a \in \{1, 3\}, x \in \{2, 3, 4, 5\}\}$. The batch size is selected from $\{32, 64, 128, 256, 512\}$. The rate for dropout is tuned from $\{0.25, 0.5, 0.75\}$. For our model, the number of hyperedges is set as 128 by default. Detailed hyper-parameter settings can be found in our released source code.

4.2 Overall Performance Comparison (RQ1)

In this section, we validate the effectiveness of our SHT framework by conducting the overall performance evaluation on the three datasets and comparing SHT with various baselines. We also re-train SHT and the best-performed baseline (*i.e.* SGL) for 10 times to compute p-values. The results are presented in Table 2.

- **Performance Superiority of SHT**. As shown in the results, SHT achieves best performance compared to the baselines under both top-*20* and top-*40* settings. The t-tests also validate the significance of performance improvements. We attribute the superiority to: i) Based on the hypergraph transformer, SHT not only realizes global message passing among semantically-relevent users/items, but also refines the hypergraph structure using the multi-head attention. ii) The global-to-local self-augmented learning distills knowledge from the high-level hypergraph transformers to regularize the topology-aware embedding learning, and thus alleviate the data noise issue.
- **Effectiveness of Hypergraph Architecture**. Among the state-of-the-art baselines, approaches that based on hypergraph neural networks (HGNN) (*i.e.*, HyRec and DHCF) outperforms most of the GNN-based baselines (*e.g.*, GCMC, PinSage, NGCF, STGCN). This sheds lights on the insufficiency of conventional GNNs in capturing high-order and global graph connectivity. Meanwhile, our SHT is configured with transformer-like hypergraph structure learning which further excavates the potential of HGNN in global relation learning. In addition, most existing hypergraph-based models utilize user or item nodes as hyperedges, while our SHT adopts latent hyperedges which not only enables automatic graph dependency modeling, but also avoids pre-calculating the large-scale high-order relation matrix.
- **Effectiveness of Self-Augmented Learning**. From the evaluation results, we can observe that self-supervised learning obviously improves existing CF frameworks (*e.g.*, MHCN, SLRec, SGL). The improvements can be attributed to incorporating the augmented learning task, which provides the beneficial regularization on the parameter learning based on the input data itself.

Table 2: Performance comparison on Yelp, MovieLens, Amazon datasets in terms of *Recall* and *NDCG*.

Data	Metric	BiasMF	NCF	AutoR	GCMC	PinSage	NGCF	STGCN	LightGCN	GCCF	DGCF	HyRec	DHCF	MHCN	SLRec	SGL	*SHT*	p-val.
Yelp	Recall@20	0.0190	0.0252	0.0259	0.0266	0.0345	0.0294	0.0309	0.0482	0.0462	0.0466	0.0472	0.0449	0.0503	0.0476	0.0526	**0.0651**	$9.3e^{-7}$
	NDCG@20	0.0161	0.0202	0.0210	0.0251	0.0288	0.0243	0.0262	0.0409	0.0398	0.0395	0.0395	0.0381	0.0424	0.0398	0.0444	**0.0546**	$9.1e^{-8}$
	Recall@40	0.0371	0.0487	0.0504	0.0585	0.0599	0.0522	0.0504	0.0803	0.0760	0.0774	0.0791	0.0751	0.0826	0.0821	0.0869	**0.1091**	$4.1e^{-7}$
	NDCG@40	0.0227	0.0289	0.0301	0.0373	0.0385	0.0330	0.0332	0.0527	0.0508	0.0511	0.0522	0.0493	0.0544	0.0541	0.0571	**0.0709**	$2.2e^{-7}$
Gowalla	Recall@20	0.0196	0.0171	0.0239	0.0301	0.0576	0.0552	0.0369	0.0985	0.0951	0.0944	0.0901	0.0931	0.0955	0.0925	0.1030	**0.1232**	$5.3e^{-7}$
	NDCG@20	0.0105	0.0106	0.0132	0.0181	0.0373	0.0298	0.0217	0.0593	0.0535	0.0522	0.0498	0.0505	0.0574	0.0581	0.0623	**0.0731**	$6.3e^{-7}$
	Recall@40	0.0346	0.0216	0.0343	0.0427	0.0892	0.0810	0.0542	0.1431	0.1392	0.1401	0.1306	0.1356	0.1393	0.1305	0.1500	**0.1804**	$1.5e^{-7}$
	NDCG@40	0.0145	0.0118	0.0160	0.0212	0.0417	0.0367	0.0262	0.0710	0.0684	0.0671	0.0669	0.0660	0.0689	0.0680	0.0746	**0.0881**	$3.2e^{-7}$
Tmall	Recall@20	0.0103	0.0082	0.0103	0.0103	0.0202	0.0180	0.0146	0.0225	0.0209	0.0235	0.0233	0.0156	0.0203	0.0191	0.0268	**0.0387**	$4.3e^{-9}$
	NDCG@20	0.0072	0.0059	0.0072	0.0072	0.0136	0.0123	0.0105	0.0154	0.0141	0.0163	0.0160	0.0108	0.0139	0.0133	0.0183	**0.0262**	$4.9e^{-9}$
	Recall@40	0.0170	0.0140	0.0174	0.0159	0.0345	0.0310	0.0245	0.0378	0.0356	0.0394	0.0350	0.0261	0.0340	0.0301	0.0446	**0.0645**	$4.0e^{-9}$
	NDCG@40	0.0095	0.0079	0.0097	0.0086	0.0186	0.0168	0.0140	0.0208	0.0196	0.0218	0.0199	0.0145	0.0188	0.0171	0.0246	**0.0352**	$3.5e^{-9}$

Specifically, MHCN regularizes the node embeddings according to a read-out global information of the holistic graph. This approach may be too strict for large graphs containing many local sub-graphs with their own characteristics. Meanwhile, SLRec and SGL adopt stochastic data augmentation to construct multiple data views, and conduct contrastive learning to capture the invariant feature from the corrupted views. In comparison to the above methods, the self-augmentation in our SHT has mainly two merits: i) SHT adopts meta networks to generate global-structure-aware mapping functions for domain adaption, which adaptively alleviates the gap between local and global feature spaces. ii) Our self-supervised approach does not depend on random masking, which may drop important information to hinder representation learning. Instead, SHT self-augment the model training by transferring knowledge from the high-level hypergraph embeddings to the low-level topology-aware embedding. The superior performance of SHT compared to the baseline self-supervised approaches validates the effectiveness of our new design of self-supervised learning paradigm.

4.3 Model Ablation Test (RQ2)

To validate the effectiveness of the proposed modules, we individually remove the applied techniques in the three major parts of SHT (*i.e.*, the local graph structure capturing, the global relation learning, and the local-global self-augmented learning). The variants are re-trained for test on the three datasets. Both prominent components (*e.g.*, the entire hypergraph transformer) and small modules (*e.g.*, the deep hyperedge feature extraction) of SHT are ablated. The results can be seen in Table 3. We have the following major conclusions:

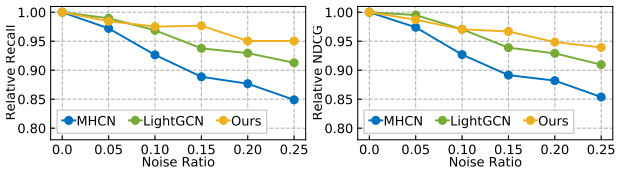
- Removing either the graph topology-aware embedding module or the hypergraph transformer (*i.e.*, *-Pos* and *-Hyper*) severely damage the performance of SHT in all cases. This result suggests the necessity of local and global relation learning, and validates the effectiveness of our GCN-based topology-aware embedding and hypergraph transformer networks.
- The variant without self-augmented learning (*i.e. -SAL*) yields obvious performance degradation in all cases, which validates the positive effect of our augmented global-to-local knowledge transferring. The effect of our meta-network-based domain adaption can also be observed in the variant *-Meta*.
- We also ablate the components in our hypergraph neural network. Specifically, we substitute the hypergraph transformer with independent node-hypergraph matrices (*-Trans*), and we remove the deep hyperedge feature extraction to keep only one layer of hyperedges (*-DeepH*). Additionally, we remove the high-order hypergraph iterations (*-HighH*). From the results we can conclude that: i) Though using much less parameters, the transformer-like hypergraph attention works much better than learning hypergraph-based user/item dependencies. ii) The deep hyperedge layers indeed make contribution to the global relation learning through non-linear feature transformation. iii) Though our hypergraph transformer could connect any users/items using learnable hyperedges, high-order iterations still improve the model performance through the iterative hypergraph propagation.

Table 3: Ablation study on key components of SHT.

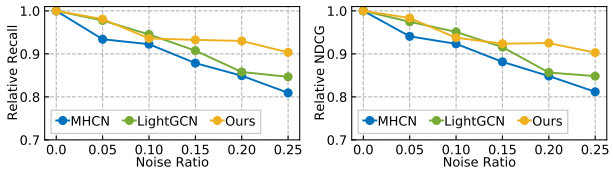
Category	Data	Yelp		Gowalla		Tmall	
	Variants	Recall	NDCG	Recall	NDCG	Recall	NDCG
Local	-Pos	0.0423	0.0352	0.0816	0.0487	0.0218	0.0247
Global	-Trans	0.0603	0.0504	0.0999	0.0608	0.0321	0.0206
	-DeepH	0.0645	0.0540	0.1089	0.0634	0.0347	0.0234
	-HighH	0.0598	0.0497	0.1091	0.0646	0.0336	0.0227
	-Hyper	0.0401	0.0346	0.0879	0.0531	0.0209	0.0144
SAL	-Meta	0.0615	0.0526	0.1108	0.0717	0.0375	0.0255
	-SAL	0.0602	0.0519	0.1099	0.0699	0.0363	0.0251
SHT		0.0651	0.0546	0.1232	0.0731	0.0387	0.0262

4.4 Model Robustness Test (RQ3)

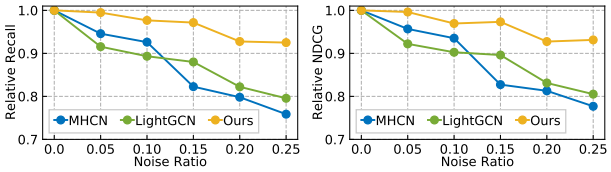
4.4.1 **Performance *w.r.t.* Data Noise Degree.** In this section, we first investigate the robustness of SHT against the data noise. To evaluate the influence of noise degrees on model performance, we randomly substitute different percentage of real edges with randomly-generated fake edges, and re-train the model using the corrupted graphs as input. Concretely 5%, 10%, 15%, 20%, 25% of the edges are replaced with noisy signals in our experiments. We compare SHT with MHCN and LightGCN, which are recent recommenders based on HGNN and GNN, respectively. To better study the effect of noise on performance degradation, we evaluate the relative performance compared to the performance on original data. The results are shown in Fig 3. We can observe that our method presents smaller performance degradation in most cases compared to the baselines. We ascribe this observation to two reasons: i) The global relation learning and information propagation by our hypergraph transformer alleviate the noise effect caused by the raw observed user-item interactions. ii) The self-augmented learning task distills knowledge from the refined hypergraph embeddings, so as to refine the graph-based embeddings. In addition, we can observe that the relative performance degradation on the Gowalla data is more obvious compared with other two datasets. This is because the noisy data has larger influence for the performance on the sparsest Gowalla dataset.



(a) Yelp data



(b) Gowalla data



(c) Tmall data

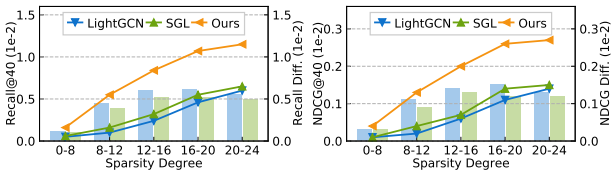
Figure 3: Relative performance degradation *w.r.t* noise ratio.

4.4.2 **Performance *w.r.t.* Data Sparsity.** We further study the influence of data sparsity from both user and item side on model performance. We compare our SHT with two representative baselines LightGCN and SGL. Multiple user and item groups are constructed in terms of their number of interactions in the training set. For example, the first group in the user-side experiments contains users interacting with 15-20 items, and the first group in the item-side experiment contains items interacting with 0-8 users.

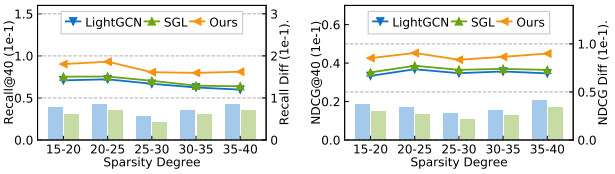
In Fig 4, we present both the recommendation accuracy and performance difference between our SHT and compared methods. From the results, we have the following observations: i) The superior performance of SHT is consistent on datasets with different sparsity degrees, which validates the robustness of SHT in handling sparse data for both users and items. ii) The sparsity of item interaction vectors has obviously larger influence on model performance for all the methods. This indicates that the collaborative pattern of items are more difficult to model compared to users, such that more neighbors usually result in better representations. iii) In the item-side experiments, the performance gap on the middle sub-datasets is larger compared to the gap on the densest sub-dataset. This suggests the better anti-sparsity capability of SHT for effectively transferring knowledge among dense samples and sparse samples with our proposed hypergraph transformer.

4.5 Case Study (RQ4)

In this section, we analyze the concrete data instances to investigate the effect of our hypergraph transformer with self-augmentation from two aspects: i) Is the hypergraph-based dependency modeling in SHT capable of learning useful node-wise relations, especially the implicit relations unknown to the training process? ii) Is the self-augmented learning with meta networks in SHT able to differentiate noisy edges in the training data? To this end, we select three users



(a) Performance *w.r.t.* item interaction numbers



(b) Performance *w.r.t.* user interaction numbers

Figure 4: Performance *w.r.t.* different data sparsity degrees on Gowalla data. Lines present Recall@40 and NDCG@40 values, and bars shows performance differences between baselines and our SHT with corresponding colors.

with fair number of interactions from Tmall dataset. The interacted items are visualized as colored circles representing their trained embeddings (refer to the supplementary material for details about the visualization algorithm). The results are shown in Fig 5. For the above questions, we have the following observations:

- **Implicit relation learning**. Even if the items are interacted by same users, their learned embeddings are usually divided into multiple groups with different colors. This may relate to users' multiple interests. To study the differences between the item groups, we present additional item-wise relations that are not utilized in the training process. Specifically, we connect items belonging to same categories, and items co-interacted by same users. Note that only view data is used in model training, so interactions in other behaviors are unknown to the trained model. It is clear that there exist dense implicit correlations among same-colored items (*e.g.*, the green items of user (a), the purple items of user (b), and the orange items of user (c)). Meanwhile, there are much less implicit relations between items of different colors. This results shows the capability of SHT in identifying useful implicit relations, which we ascribe to the global structure learning of our hypergraph transformer.
- **Noise discrimination**. Furthermore, we show the solidity scores s estimated from our self-augmented learning, for the user-item relations in Fig 5. We also show the normalized values of some notable edges in the corresponding circles (*e.g.*, edges of item 10202 and 6508 are labeled with 2.3 and 1.9). The red values are anomalously low, which may indicates noise. The black values are the lowest and highest solidity scores for edges except the anomalous ones. By analyzing user (a), we can regard the yellow and green items as two interests of user (a) as they are correlated in terms of their learned embeddings. In contrast, item 6508 and 10202 have few relations to other interacted items of user (a), which may not reflect the real interactive patterns of this user. Thus, the model may consider this two edges as noisy interactions. Similar cases can be found for user (b), where item 2042 has few connections to the other items and show difference with the

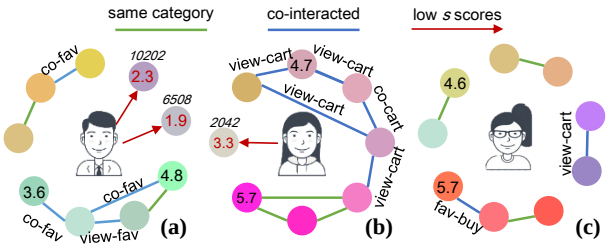


Figure 5: Case study on inferring implicit item-wise relations and discriminating potential noise edges. Circles denote items interacted by the centric users, and their learned embeddings are visualized with colors. Implicit item-wise relations not utilized during model training are presented by green and blue lines. The type of co-interactions are also labeled (*e.g.*, *view-cart* denotes viewed and added-to-cart by same users). Also, the inferred solidity scores s are shown on the circles, where red values are anomalously low scores indicating noisy edges.

embedding color. It is labeled with low s scores and considered as noise by SHT. The results show the effective noise discrimination ability of the self-augmented learning in SHT, which recalibrate the topology-aware embedding using global information encoded from hypergraph transformer.

5 CONCLUSION

In this work, we explore the self-supervised recommender systems with an effective hypergraph transformer network. We propose a new recommendation framework SHT, which seeks better user-item interaction modeling with self-augmented supervision signals. Our SHT model improves the robustness of graph-based recommender systems against noise perturbation. In our experiments, we achieved better recommendation results on real-world datasets. Our future work would like to extend our SHT to explore the disentangled user intents with diverse user-item relations for encoding multi-dimensional user preferences.

ACKNOWLEDGMENTS

This research work is supported by the research grants from the Department of Computer Science & Musketeers Foundation Institute of Data Science at the University of Hong Kong.

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In *KDD*.
[2] 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 *SIGIR*. 335–344.
[3] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*, Vol. 34. 27–34.
[4] Yudong Chen, Xin Wang, Miao Fan, Jizhou Huang, Shengwen Yang, et al. 2021. Curriculum meta-learning for next POI recommendation. In *KDD*. 2692–2702.
[5] Ruocheng Guo, Xiaoting Zhao, Adam Henderson, Liangjie Hong, and Huan Liu. 2020. Debiasing grid-based product search in e-commerce. In *KDD*. 2852–2860.
[6] Xiangnan He, Kuan Deng, Xiang Wang, et al. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.
[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[8] Chao Huang. 2021. Recent advances in heterogeneous relation learning for recommendation. *arXiv preprint arXiv:2110.03455* (2021).
[9] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xiao, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *AAAI*.
[10] Dasol Hwang, Jinyoung Park, Sunyoung Kwon, KyungMin Kim, Jung-Woo Ha, and Hyunwoo J Kim. 2020. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. *NIPS* 33 (2020), 10294–10305.
[11] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In *KDD*. 2020–2029.
[12] Minguk Kang and Jaesik Park. 2020. Contragan: Contrastive learning for conditional image generation. *NIPS* 33 (2020), 21357–21369.
[13] Yehuda Koren, Robert Bell, et al. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[14] Adit Krishnan, Ashish Sharma, et al. 2018. An adversarial approach to improve long-tail performance in neural collaborative filtering. In *CIKM*. 1491–1494.
[15] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.
[16] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *TKDE* (2021).
[17] Yiyu Liu, Qian Liu, Yu Tian, et al. 2021. Concept-Aware Denoising Graph Neural Network for Micro-Video Recommendation. In *CIKM*. 1099–1108.
[18] Zhen Peng, Wenbing Huang, Minnan Luo, et al. 2020. Graph representation learning via graphical mutual information maximization. In *WWW*. 259–270.
[19] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. In *SIGIR*. 89–98.
[20] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Recsys*. 240–248.
[21] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.
[22] 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 *CIKM*. 1441–1450.
[23] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax.. In *ICLR*.
[24] Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2021. LexFit: Lexical fine-tuning of pretrained language models. In *ACL*. 5269–5283.
[25] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *SIGIR*. 1101–1110.
[26] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.
[27] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.
[28] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *SIGIR*. 1001–1010.
[29] Yifan Wang, Suyao Tang, et al. 2020. Disenhan: Disentangled heterogeneous graph attention network for recommendation. In *CIKM*. 1605–1614.
[30] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, et al. 2019. Simplifying graph convolutional networks. In *ICML*. PMLR, 6861–6871.
[31] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, et al. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.
[32] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *AAAI*, Vol. 35. 4486–4493.
[33] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Xiangji Huang. 2022. Hypergraph Contrastive Collaborative Filtering. *arXiv preprint arXiv:2204.12200* (2022).
[34] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. 2021. Graph meta network for multi-behavior recommendation. In *SIGIR*. 757–766.
[35] Haoran Yang, Hongxu Chen, Lin Li, et al. 2021. Hyper Meta-Path Contrastive Learning for Multi-Behavior Recommendation. In *ICDM*. IEEE, 787–796.
[36] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge Graph Contrastive Learning for Recommendation. *arXiv preprint arXiv:2205.00976* (2022).
[37] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, et al. 2021. Self-supervised Learning for Large-scale Item Recommendations. In *CIKM*. 4321–4330.
[38] Rex Ying, Ruining He, Kaifeng Chen, et al. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.
[39] 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 *WWW*. 413–424.
[40] Jiani Zhang, Xingjian Shi, Shenglin Zhao, et al. 2019. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. In *IJCAI*.
[41] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A model of two tales: Dual transfer learning framework for improved long-tail item recommendation. In *WWW*. 2220–2231.
[42] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, et al. 2021. Causal intervention for leveraging popularity bias in recommendation. In *SIGIR*. 11–20.

6 SUPPLEMENTAL MATERIAL

In the supplementary material, we first show the learning process of SHT with pseudocode summarized in Algorithm 1. Then, we investigate the influences of different hyperparameter settings, and discuss the impact of three key hyperparameters. Finally, we describe the details of our vector visualization algorithm used in the case study experiments.

6.1 Learning process of SHT

Algorithm 1: Learning Process of SHT Framework

Input: user-item interaction graph $\mathcal{G}$, number of graph layers L, number of edges to sample R, R', maximum epoch number E, learning rate η
Output: trained parameters in Θ

Initialize all parameters in Θ
for $e = 1$ *to* E **do**
  Draw a mini-batch $\mathbf{U}$ from all users $\{1, 2, ..., I\}$
  Calculate the graph topology-aware embeddings $\bar{\mathbf{E}}$
  Generate input embeddings $\tilde{\mathbf{E}}_0$ for hypergraph transformer
  for $l = 1$ *to* L **do**
    Conduct node-to-hyperedge propagation to obtain $\tilde{\mathbf{Z}}^{(u)}, \tilde{\mathbf{Z}}^{(v)}$ for both users and items
    Conduct hierarchical hyperedge feature transformation for $\hat{\mathbf{Z}}^{(u)}, \hat{\mathbf{Z}}^{(v)}$
    Propagate information from hyperedges back to user/item nodes to obtain $\tilde{\mathbf{E}}_l^{(u)}, \tilde{\mathbf{E}}_l^{(v)}$
  end
  Aggregate the iteratively propagated embeddings to get $\hat{\mathbf{E}}$
  Sample R edge pairs for self-augmented learning
  Acquire the user/item transformation function $\phi^{(u)}$ and $\phi^{(v)}$ with the meta network
  Conduct user/item embedding transformations using $\phi(\cdot)$ to get $\mathbf{\Gamma}^{(u)}, \mathbf{\Gamma}^{(v)}$
  Calculate the solidity score s for the R edge pairs
  Calculate the solidity predictions $\hat{s}$ for the R edge pairs
  Compute loss $\mathcal{L}_{\text{sa}}$ for self-augmented learning according to Eq 12
  Sample R' edge pairs for the main task
  Calculate the pair-wise marginal loss $\mathcal{L}$ according to Eq 13
  for *each parameter* $\theta \in \Theta$ **do**
    $\theta = \theta - \eta \cdot \partial\mathcal{L}/\partial\theta$
  end
end
return all parameters Θ

6.2 Hyperparameter Investigation

We study the effect of three important hyperparameters, *i.e.*, the hidden dimensionality d, the number of latent hyperedges K, and the number of graph iterations L. To present more results, we calculate the relative performance decrease in terms of evaluation metrics, compared to the best performance under default settings. The results are shown in Fig 6, our observations are shown below:

- The latent embedding dimension size largely determines the representation ability of the proposed SHT model. Small d greatly limits the efficacy of SHT, by 15%-35% performance decrease. However, greater d does not always yield obvious improvements. As shown by results when $d = 68$ on Yelp data, the performance increases marginally due to the over-fitting effect.
- The curve of performance w.r.t. hyperedge number K typically follows the under- to over-fitting pattern. However, it is interesting that K has significantly less influence compared to d (at most −6% and −35%, respectively). This is because the hyperedge-node connections in SHT are calculated in a d-dimensional space, which reduces the amount of independent parameters related to K to $O(K \times d)$. So K has much smaller impact on model capacity compared to d, which relates to $O((I + J) \times d)$ parameters.
- For the number of graph iterations L, smaller L hinders nodes from aggregating high-order neighboring information. When $L = 0$, graph neural networks degrades significantly. Meanwhile, by stacking more graph layers may cause over-smoothing issue, which yields indistinguishable node embeddings.

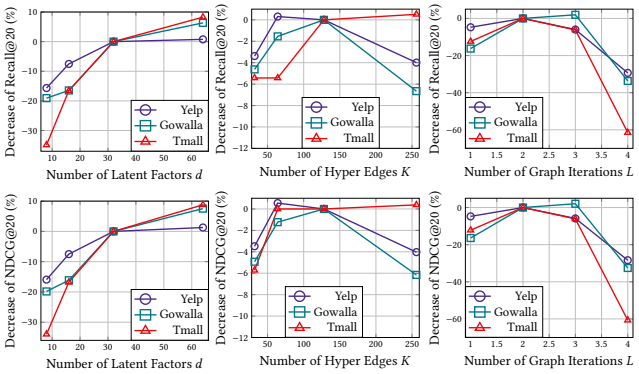


Figure 6: Hyperparameter study of the SHT.

6.3 Vector Visualization Algorithm

In our case study experiments, each item embeddings of 32 dimensions is visualized with a color. This visualization process should preserve the learned item information in the embedding vectors. Meanwhile, to make the visualization results easy to understand, it would be better to pre-select several colors to use. Considering the above two requirements, we design a neural-network-based dimension reduction algorithm. Specifically, we train a multi-layer perceptron to map 32-dimensional item embeddings to 3-dimensional RGB values. The network is trained using two objective functions, corresponding to the forgoing two requirements. Firstly, the compressed 3-d vectors (colors) are fed into a classifier, to predict the original item ids. Through this self-discrimination task, the network is trained to preserve the original embedding information in the RGB vectors. Secondly, the network is trained with a regularizer that calculates the distance between each color vectors and the preferred colors. Using the two objectives, we can map embeddings into preferred colors while preserving the embedding information.