Self-supervised Hypergraph Transformer with Alignment and Uniformity for Recommendation

XianFeng Yang, Yang Liu

Abstract—**Graph neural networks have proven their effectiveness for user-item interaction graph collaborative filtering. However, most of the existing recommendation models highly depended on abundant and high-quality datasets and neglected the alignment and uniformity of embedding representation. The noisy and skewed distribution of data from real-world applications is ubiquitous. The alignment and uniformity are crucial for representation learning. In this paper, we propose SHTAU (self-supervised hypergraph transformer with alignment and uniformity for the recommendation) to address the problem. Specifically, we first perform a graph neural network on the user-item interaction graph, then captures the global collaboration relationship between users and items through the hypergraph transformer structure, and use the extracted global information to generate self-supervised signals for data enhancement on the user-item interaction graph to enhance the robustness of the recommendation algorithm. Simultaneously, alignment and uniformity auxiliary task update embeddings from different angles to improve performance. Extensive experiments are conducted on Yelp, Gowalla, MovieLens, Amazon-book and Tmall datasets. The experimental results show that SHTAU has obvious advantages over the baseline methods.**

Index Terms—**Recommendation, Self-Supervised Learning, Hypergraph, Alignment and Uniformity**

I. Introduction

Recommender systems play critical roles in various applications and greatly affect the user experience. To assist users in finding what they need, personalized recommendation systems widely adopt collaborative filtering (CF) techniques. The fundamental concept behind collaborative filtering (CF) is that similar users tend to have similar preferences. Collaborative filtering can utilize the user-item interaction history to discover similar users and generate recommendations for specific users based on this similarity relation.

Earlier CF models employed matrix factorization (MF) to project interaction data into latent user and item embeddings [1]. As deep learning gained popularity, CF models leveraging neural networks started to emerge, such as NCF [2] and AutoR [3]. Recent years have witnessed notable advancements in the development of graph neural networks (GNNs) specifically designed to model graph structure data [4,5]. The technique of recursively aggregating information along user-item interactions to obtain node embeddings has demonstrated its effectiveness in practice. For example, GCMC [6] and STGCN [7] contain the GCN structure that sense and aggregate neighbor information in the user-item interaction graph. To simplify graph-based information propagation, LightGCN [8] removes redundant nonlinear transformers during information propagation and acquired better performance. In order to further improve the recommendation models based on graph neural networks, some studies suggest using disentangled graph neural frameworks to learn intent-aware representations (e.g., DisenHAN [9]) and using multi-relational graph neural models to distinguish user behavior and generate behavior-aware embeddings (e.g., MB-GMN [10]).

Existing recommendation models based on graph neural networks face the following challenges:

- **Existing graph neural network recommendation models have a high dependency on sufficient and high-quality training data**. However, noisy data is common in many recommendation scenarios, and that prevents the fulfillment of this essential need. For example, popular items could be over-recommending by the recommendation system, in this situation, users may click on products they are not interested in [11]. As a result, the user-item interaction graph could include edges that are not correlated with user interests. Ignoring the existence of these interest-uncorrelated edges and treating all edges equally will harm the accuracy of user representation. Taking a step further, when it comes to interest-uncorrelated information, recursively executing multi-hop information propagation will spread the misleading information out and amplify the influence of noise, eventually dragging the performance down.
- **Data sparsity and skewed distribution misleads GNN recommendation models and cause the recommendation to lean toward popular items [12,13]**. Therefore, important training signals could be relatively weak, making it harder for the model to discover them and process them appropriately. Although there are some recently developed recommendation models which utilize self-supervised learning to optimize user representation. Most of them use a random mask strategy that may preserve noisy interactions or lose some important training signals during data augmentation.
- **Most recommendation models that are based on graph neural networks mainly emphasize the development of better encoders to acquire improved embedding representations of users and items**. However, they often neglect the aligning and uniformity of the embedded representations of users and items within collaborative filtering.

To address the aforementioned challenges, this study proposes the Self-supervised Hypergraph Transformer with

Manuscript received September 25, 2023; revised January 11, 2024.

XianFeng Yang is a postgraduate student at School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China (e-mail: 1215867299@qq.com).

Yang Liu is a professor at School of computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China (corresponding author to provide e-mail: liuyang_lnas@163.com).

Alignment and Uniformity for recommendation framework (SHTAU) to enhance the robustness and generalization performance of the collaborative filtering recommendation model.

Specifically, this paper enhances the capabilities of the model by utilizing self-supervised signals, hypergraph transformers, and alignment-uniform learning auxiliary tasks.

The primary contributions of this paper can be summarized as follows:

- SHTAU, a recommendation framework based on a self-supervised hypergraph transformer and alignment uniformity is proposed to mitigate the negative impact of noise.
- Considering the target properties of user and item embedding representations in collaborative filtering, the alignment-uniform learning objective is introduced to enhance the performance of the model.
- A large number of experiments are carried out on five public datasets, and the experimental results show that the performance of the proposed SHTAU framework is better than baseline methods.

II. Related Work

The related works in this paper include recommendation algorithms with graph neural networks, recommendation algorithms with hypergraphs, and recommendation algorithms with self-supervised learning.

A. Recommendation with graph neural network

Recent recommender system research has designed a variety of graph neural network structures to learn the information contained in user-item interactions. For example, PinSage [14] and NGCF [15] propose graph convolutional networks built on spectral domains. LightGCN [8] proposes a simplified GCN for collaborative filtering and improves performance by using sum-based pooling on interactions. In the GCN-based model structure, every user and item is encoded into an embedding by aggregating information along the edges of the user-item interaction graph. In order to refine user embedding, several disentangled graph neural network architectures are proposed, including DGCF [16] and DisenHAN [17]. Despite more effective CF models are emerging, most of them rely only on observed interactions for model training, which makes them ineffective in modeling interactions when facing sparse and noisy signals. In order to overcome these challenges, this work generates higher-order knowledge by self-supervised learning structure from hypergraph information to filter noises and to focus on learning effective interactions.

B. Recommendation with hypergraph

Compared to ordinary graphs, hypergraphs can provide a more accurate representation of the relationships between nodes with multiple associations. Hypergraphs can represent complex multivariate relationships and higher-order relationships [26]. Some recent hypergraph-based models have improved user-item relational learning for recommender systems. DHCF [18] is a hypergraph collaborative filtering model to learn the hybrid high-order correlations. Where HyRec [19] treats the user as a series of hyperedges that aggregate information from related nodes. MHCN [20] is a multi-channel hypergraph convolutional network to enhance social recommendation by leveraging high-order user relations. Unlike the above models, SHTAU automatically learns hypergraph structures through global collaboration relationships instead of designing them manually.

C. Contrastive learning using alignment and uniform

Recent studies [21, 22] in the field of contrastive learning identify alignment and uniformity as two key properties that can be used to efficiently build high-quality representations. Given the distribution of data $p_{data}(\cdot)$ and the distribution of positive pairs $p_{pos}(\cdot)$, alignment is defined as the average distance between normalized embeddings of positive pairs:

$$\mathcal{L}_{align} \triangleq \mathop{\mathbb{E}}_{(x,x^{+})\sim p_{pos}} \|f_{NR}(x) - f_{NR}(x^{+})\|^2 \qquad (1)$$

Where $f_{\mathrm{NR}}(\cdot)$ indicates l_2 normalized representations. And we define the uniformity loss as the logarithm of the average pairwise Gaussian potential:

$$\mathcal{L}_{uniform} \triangleq \mathop{\mathbb{E}}_{(x,y)\sim p_{data}} e^{-2\|f_{NR}(x)-f_{NR}(y)\|^2} \qquad (2)$$

The ideal embedding representation is highly consistent with the state described by these two objective functions, i.e., the user and item representations of a pair of positive samples should be close to each other, while the representations of randomly selected samples should be as far away from each other as possible, and scattered throughout the embedding space. In this study, these two additional learning objectives will be incorporated into multi-objective learning to enhance the embedding's quality.

III. Symbol Definitions

This section explains the definitions of symbols used in this work.

TABLE I
Symbols and Their Definitions

Symbol	Definitions
G	User-item interaction graph
A	The adjacency matrix of the graph
$\bar{A}$	Normalized adjacency matrix
D	Degree matrix of graph
$\hat{E}$	Global user/item embedding
E_{local}	Local user/item embedding
$\hat{Z}$	Hyperedge embedding
$\bar{Z}$	Multi-head hyperedge embedding
q, k, v	Transformer parameter matrices
Γ	User/item solidity embedding
s	Edge solidity label
$\hat{s}$	Edge solidity estimate

Definition 1. Collaborative Filtering. Let U and I denote the user and item set, respectively. Given a set of observed user-item interactions R = $\{(u,i) \mid u$ interacted with $i\}$, CF methods aim to infer the score $s(u,i) \in$ R for each unobserved user-item pair which indicates how likely the user u tends to interact with the item i.

IV. Methodology

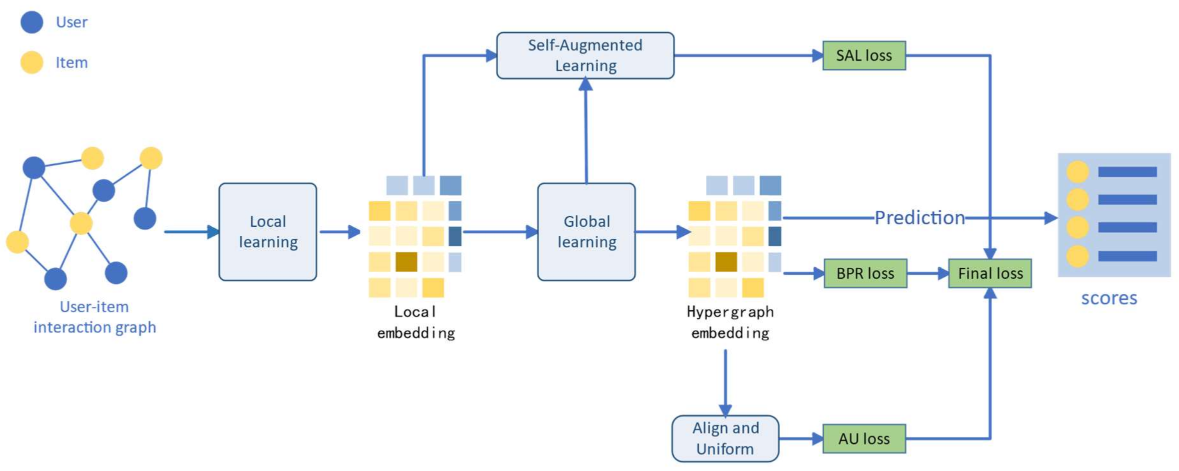


Fig.1 Complete framework of SHTAU

In this section, we elaborate on the architecture of the SHTAU framework. The complete model architecture is illustrated in Figure 1. The SHTAU framework structure mainly includes the following components: graph neural network for local learning, hypergraph encoder for global embedding, local-global self-augmented learning, and align-uniform learning.

A. Local graph structure learning

We initialize user and item representations in a d-dimensional latent space, for user u_i and item v_j we get embedding vectors $e_i, e_j \in \mathbb{R}^d$. Stack all embedding vectors into embedding matrices $E^{(u)} \in \mathbb{R}^{I\times d}$, $E^{(v)} \in \mathbb{R}^{J\times d}$. To perform the local embedding, a two-layer Graph Convolutional Network (GCN) is employed. In addition to the graph convolutional layers, residual connections are utilized in the model architecture. Residual connections facilitate the flow of information through the network by allowing the model to retain and propagate information from previous layers. This helps to mitigate the problem of vanishing gradients and can improve the training process.

$$E^{(u)}_{local} = GCN^2\left(E^{(v)}, G\right) = \bar{A}\cdot\bar{A}^T E^{(u)} + \bar{A}\cdot E^{(v)} \tag{3}$$

Where $E^{(u)}_{local} \in \mathbb{R}^{I\times d}$ denotes user side local embedding. $\bar{A} \in \mathbb{R}^{I\times J}$ denotes the normalized adjacency matrix. Where G represents the user-item interaction graph

$$\bar{A}_{i,j} = A_{i,j} / \left(D^{(u)1/2}_i D^{(v)1/2}_j\right) \tag{4}$$

$D^{(u)1/2}_i$, $D^{(v)1/2}_j$ denotes the degree of node u_i, v_j in graph G. Item side local embedding calculated likewise.

B. Hypergraph global learning

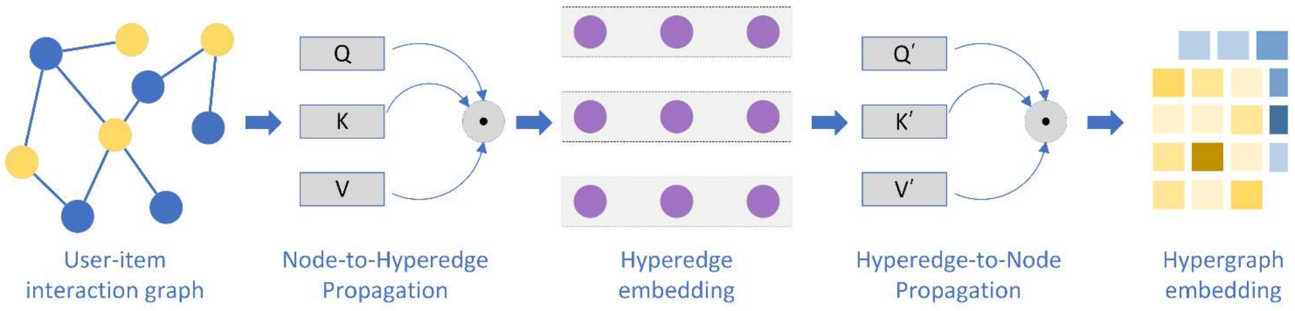
Despite graph neural networks have shown their effectiveness in recommender system, The model's performance is still restricted by the sparse data problem. To overcome this limitation, a hyper-graph encoder is employed, which aims to extract and integrate global information by leveraging a larger receptive field. By utilizing a hyper-graph structure, the model can capture and propagate information across a broader context, enabling it to better understand the relationships and dependencies in the data. The structure of the hypergraph encoder is shown in Figure 2.

a) Node to hyperedge propagation

This section elaborates on the propagation process of embedding from user nodes to user-side hyperedges. The same processing method applies to the item side. The objective is to effectively capture and preserve important information during the propagation, rather than letting it fade away in multiple hypergraph propagations. To achieve this, a transformer-like structure is utilized. Transformers have demonstrated their effectiveness in capturing and leveraging long-range dependencies in various tasks, including natural language processing and computer vision. By adopting a transformer-like structure, the model aims to capture valid information directly from the user nodes and incorporate it into the hyperedge embeddings. The procedures of node-hyperedge propagation can be expressed as follows:

$$\tilde{z}_k = \left(\bar{z}_{k,1}, \bar{z}_{k,2}, \ldots, \bar{z}_{k,h}\right); \bar{z}_{k,h} = \sum_{k=1}^{K} v_{k,h}\, k^T_{k,h} q_{i,h} \tag{5}$$

Where $\tilde{Z}_k \in R^d$ represents the embedding of the k-th hyperedge. Concatenated with H multi-head embeddings



Q
K
V
Q′
K′
V′
User-item interaction graph
Node-to-Hyperedge Propagation
Hyperedge embedding
Hyperedge-to-Node Propagation
Hypergraph embedding

Fig.2 Hypergraph encoder for global learning

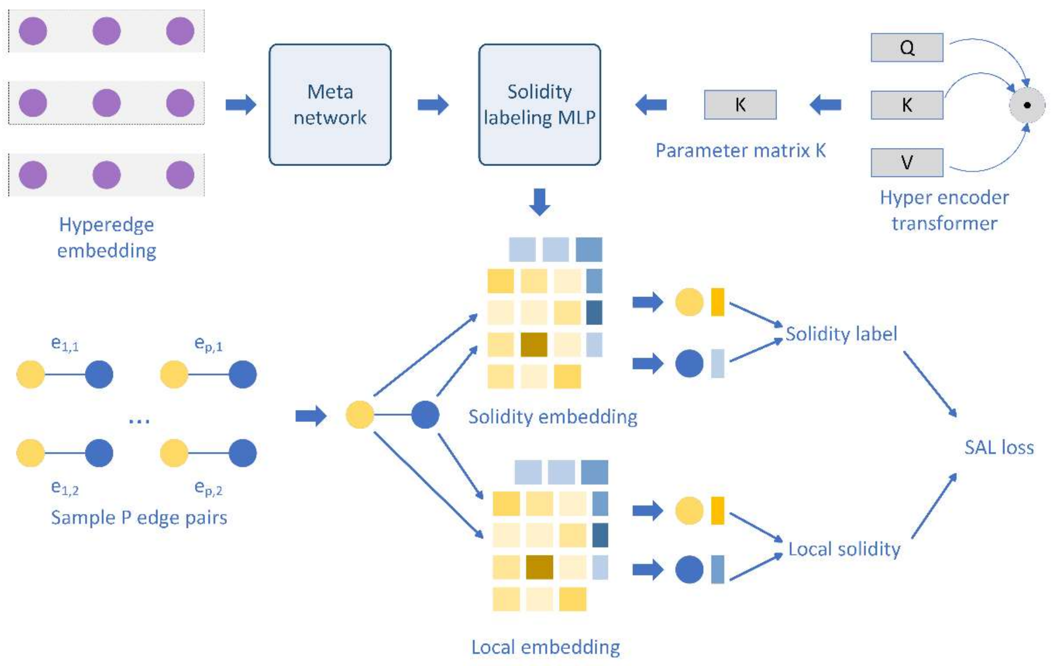


Fig.3 Self-Augmented Learning

$\bar{Z}_{k,h} \in \mathbb{R}^{d/H}$.

$$q_{k,h} = Z_{k,p_{h-1}:p_h}; k_{i,h} = K_{p_{h-1}:p_h,:}\tilde{e}_i; v_{i,h} = V_{p_{h-1}:p_h,:}\tilde{e}_i \quad (6)$$

Where $q_{k,h}, k_{i,h}, v_{i,h} \in \mathbb{R}^{d/H}$ represents the query, key, value in attention mechanism respectively. $Z \in R^{K\times d}$ represents the embedding matrix of the K hyperedge. The final hyperedge embedding can be obtained as follows:

$$\hat{Z} = HHGN^2(\tilde{Z}); HHGN(\boldsymbol{X}) = \sigma(H \cdot X + X) \quad (7)$$

Where $\hat{Z}$ represents the final hyperedge embedding. $\tilde{Z} \in \mathbb{R}^{K\times d}$ represents the output of transformer. $\mathrm{HHGN}^2(\cdot)$ represents performing the hierarchical hypergraph network two times consecutively. $H \in \mathbb{R}^{K\times K}$ represents the parameter matrix of HHGN, it characterizes relationship between hyperedges. $\sigma(\cdot)$ denotes an activation function.

b) Hyperedge to node propagation

To get distilled information into use we need a hyperedge-node propagation. In node-hyperedge propagation, information flows from individual nodes to hyperedges, allowing the hyperedges to capture and aggregate the information from their neighboring nodes. Conversely, in the hyperedge-node propagation, the distilled information is propagated back from hyperedges to individual nodes. This propagation process enables the nodes to receive the aggregated information from the hyperedges and update their representations accordingly.

$$\tilde{\boldsymbol{e}}'_i = (\bar{e}'_{i,1}, \bar{e}'_{i,2}, \dots, \bar{e}'_{i,h}); \quad \bar{e}'_{i,h} = \sum_{i=1}^{I} v'_{i,h} {k'}^{\top}_{k,h} q'_{k,h} \quad (8)$$

Where $\tilde{e}'_i \in \mathbb{R}^d$ means the hypergraph embedding of user u_i that generated by hypergraph encoder $\bar{e}'_{i,h}$ denotes the h-th multi-head embedding of user u_i.

$$q'_{i,h} = k_{i,h}; k'_{i,h} = q_{i,h}; v'_{k,h} = V_{p_{h-1}:p_h,:}\hat{z}_k \quad (9)$$

This process shares parameters with note-hyperedge propagation, the former key now is the new query and the former query is the new key. $\hat{z}_k$ denotes the embedding of k-th hyperedge.

c) Recursively hypergraph propagation

To enhance the receptive field and capture long-span global dependencies, hypergraph propagation is employed recursively. This approach allows information to propagate through a hypergraph structure, enabling the model to gather knowledge from a broader context. By recursively performing hypergraph propagation, the model can gradually incorporate information from distant nodes, resulting in a larger receptive field. This expanded receptive field helps the model to better understand and leverage global relationships and dependencies within the data. The final global embedding $\hat{E}$ can be obtained by summing the embeddings from each layer:

$$\hat{E} = \sum_{l=1}^{L} \tilde{E}_l; \ \tilde{E}_l = HyperEndcoder(\tilde{E}_{l-1}) \quad (10)$$

Where HyperEncoder(·) denotes the complete hypergraph global learning process above.

C. Self-Augmented Learning

To further eliminate the influence of noise in user-item interaction information on graph topology embedding. Introduce self-augmented learning between graph topological embeddings and hyperedge learning. Specifically, an additional task is added to the graph topology embedding information extraction process, which distinguishes the solidity of user-item interaction. The solidity of an edge in the user-item interaction graph represents the probability that the edge is not noisy. Lower solidity values indicate a higher likelihood of noise in the edge. By calculating the solidity labeling, the framework can quantify the noise levels in the user-item interactions based on the learned hypergraph dependency representation.

a) Solidity labeling with meta network

To distinguish noisy data and leverage edge solidity for guiding embedding learning, specific calculations are needed to generate self-supervised labels based on the solidity concept. These calculations involve using the learned hypergraph dependency representation through meta-networks. The meta-network utilizes the learned hypergraph dependency representation as input to construct a perceptron. The perceptron is a type of neural network that can learn to estimate the solidity of each edge in the user-item interaction graph. By leveraging the hypergraph dependency representation, the perceptron can assess the solidity of each edge, indicating the likelihood of noise in the data. The generated solidity labels serve as self-supervised labels, guiding the embedding learning process. By incorporating these labels into the training process, the model can effectively denoise the observed user-item interaction data. The hypergraph dependency representation, being leveraged as useful knowledge, helps the model prioritize reliable interactions and reduce the impact of noise. Specifically, the parameter matrix K from the hypergraph Transformer will be reused here since they are generated for modeling relationships, and the information they contain can guide the estimation of the solidity of user-item interactions.

$$\Gamma_i = \phi^{(u)}\left(\Vert_{h=1}^{H} k_{i,h}\right); \Gamma_j = \phi^{(v)}\left(\Vert_{h=1}^{H} k_{j,h}\right) \quad (11)$$

where Γ_i , Γ_j denotes user and item solidity embedding. $\phi^{(u)}(\cdot)$, $\phi^{(v)}(\cdot)$ denotes user and item side perceptron. The project of perceptron guided by a meta network, this meta network takes user and item side hyperedge embedding as input, and output the parameter of perceptron.

$$\phi(x; Z) = \sigma(Wx + b);\ W = V_1\bar{z} + W_0; b = V_2\bar{z} + b_0 \quad (12)$$

Where $x \in \mathbb{R}^d$ is the input of the perceptron (e.g., Γ_i , Γ_j). Z is the hyperedge embedding from user-side or item-side. $W \in \mathbb{R}^{d\times d}$ and $b \in \mathbb{R}^d$ is the parameters of perceptron generated by the meta network. $\bar{z} \in \mathbb{R}^d$ is the mean pooling of hyperedge embedding. $V_1 \in \mathbb{R}^{d\times d\times d}$ $W_0 \in \mathbb{R}^{d\times d}$ $V_2 \in \mathbb{R}^{d\times d}$ $b_0 \in \mathbb{R}^d$ are parameters of the meta network.

$$s_{i,j} = sigm\left(d^{\top} \cdot \sigma\left(T \cdot [\Gamma_i; \Gamma_j] + \Gamma_i + \Gamma_j + c\right)\right) \quad (13)$$

Where $s_{i,j}$ denotes the solidity of edge (u_i, v_j). $sigm(\cdot)$ denotes the sigmoid function. $d \in \mathbb{R}^d$,$T \in \mathbb{R}^{d\times 2d}$,$c \in \mathbb{R}^d$ are parameter matrices. $[\cdot;\ \cdot]$ represents vector concatenation.

b) Self-Augmented learning objective function

To enable the framework to handle interactions with different solidities differently, a self-supervised objective function is introduced. In order to guide the optimization process of user and item embeddings, both the global solidities obtained from the aforementioned process and the local solidities predicted by the framework based on local information are needed. By having both the global and local solidities, the framework can handle interactions with different solidities differently during the optimization process of user and item embeddings. This function performs contrastive learning of local embedding solidity estimation and solidity labels:

$$\mathcal{L}_{sa} = \sum_{p=1}^{P} max\,(0,\ 1 - (\hat{s}_{u_{p,1},v_{p,1}} - \hat{s}_{u_{p,2},v_{p,2}}) \cdot (s_{u_{p,1},v_{p,1}} - s_{u_{p,2},v_{p,2}})) \quad (14)$$

$$\hat{s}_{u_{p,1},v_{p,1}} = e_{u_{p,1}}^{\top} e_{v_{p,1}}^{\top};\ \hat{s}_{u_{p,2},v_{p,2}} = e_{u_{p,2}}^{\top} e_{v_{p,2}}^{\top} \quad (15)$$

Where $\mathcal{L}_{sa}$ represents the loss function of Self-Augmented Learning. $\hat{s}$ denotes the local embedding estimate solidity. Where s denotes the solidity label. 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. By adapting the gradients based on the similarity of solidity labels, the model can focus more on refining the embeddings for edges with larger differences in solidity. This differential treatment enables the model to allocate more attention and resources to instances that are more likely to be noisy or unreliable, improving its ability to handle noisy data. The ability to differentiate between different solidities helps the model prioritize and focus on more reliable interactions while downplaying the influence of potentially noisy or unreliable interactions. By doing so, the model can improve its performance and robustness in handling real-world data with various levels of noise.

D. Alignment and uniformity auxiliary task

In recent research on recommendation systems, there has been a strong emphasis on designing complex encoders to capture intricate patterns and relationships in the data. However, less attention has been given to the desired properties of user and item representations. The alignment loss pushes up the similarity between representations of positive-related user-item pairs, while the uniformity loss measures how well the representations scatter on the hypersphere. Ideally, the representations of a pair of positive samples should exhibit similarity, and each representation should convey as much information as possible. To achieve these goals, two learning objectives can be considered:

$$\mathcal{L}_{align} \triangleq \mathop{\mathbb{E}}_{(u,i)\sim p_{pos}} \Vert f_{NR}(u) - f_{NR}(i)\Vert^2 \quad (16)$$

$$\mathcal{L}_{uniform} = \mathop{\mathbb{E}}_{(u,u')\sim p_{user}} \Vert f_{NR}(u) - f_{NR}(u')\Vert^2/2 + \mathop{\mathbb{E}}_{(i,i')\sim p_{item}} \Vert f_{NR}(i) - f_{NR}(i')\Vert^2/2 \quad (17)$$

Where $f_{\mathrm{NR}}(\mathrm{u})$ and $f_{\mathrm{NR}}(\mathrm{i})$ indicates l_2 normalized user and item representations. Align loss is expected distance between positive user and item pair. And we define the uniformity loss as the logarithm of the average pairwise Gaussian potential. Minimizing loss function $\mathcal{L}_{align}$ can make the embeddings of positive sample pairs tend to be similar. The minimization of loss function $\mathcal{L}_{uniform}$ first prevents trivial solutions where all embeddings become identical, which would excessively cater to objective function $\mathcal{L}_{align}$. Secondly, it enables embeddings to be more evenly distributed in the latent space, thus expressing more information. The structure of the alignment and uniform learning is shown in Figure 4, and the embedding of each pair of positive samples is calculated in each iteration, and the uniformity calculation is performed on the embedding on the item side and the user side respectively.

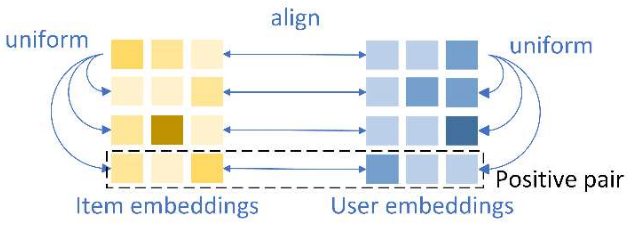


Fig.4 Alignment and uniformity learning

E. Model learning

The training of the model is accomplished by optimizing a series of objective functions, including the main task objective function, the self-augmented learning task objective function, and the alignment and uniformity objective function. The final loss is expressed as:

$$\mathcal{L} = \sum_{r=1}^{R} max\left(0,1 - \left(p_{u_{r,1},v_{r,1}} - p_{u_{r,2},v_{r,2}}\right)\right) + \lambda_1 \mathcal{L}_{sa} + \lambda_2 \mathcal{L}_{align} + \lambda_3 \mathcal{L}_{uniform} + \lambda_4 \|\Theta\|_F^2 \tag{18}$$

During training, sample r positive edges (in graph G) and sample r negative edges (not in graph G) form$\{(e_{1,1}, e_{1,1}), (e_{2,1}, e_{2,1}), \ldots, (e_{R,1}, e_{R,1})\}$, where $e_{r,1}$ and $e_{r,2}$ are positive and negative samples, respectively. $p_{u_{r,1},v_{r,1}}$ and $p_{u_{r,2},v_{r,2}}$ represent the score predictions of edge $e_{r,1}$ and $e_{r,2}$, respectively.

V. EXPERIMENTS

A. Experimental datasets

The experiments use the Yelp, Gowalla, MovieLens, Amazon-book and Tmall datasets, which are all collected from real-world applications. The statistics of them are shown in Table 2.

- Yelp: This dataset contains Yelp users' ratings of merchants, including restaurants, shopping malls, hotels, and more. Among them, the places that have been evaluated by users are used as interactive items, and all places that have not been evaluated are regarded as non-interactive items.
- Gowalla: This dataset contains the user's check-in data in the geographical location through Gowalla, and the user shares his check-in locations through Gowalla.
- Tmall: This e-commerce dataset contains the online shopping behavior of Tmall's users.
- MovieLens: It is a movie recommendation dataset.
- Amazon-book: this dataset records user ratings on products with book category on Amazon with the 20-core setting.

TABLE II
STATISTICS OF DATASETS

Stat.	Yelp	Gowalla	Tmall	MovieLens	Amazon
Number of users	29601	50821	47939	69878	78578
Number of items	24734	24734	41390	10196	77801
Number of interactions	1517326	1069128	2357450	9988816	3190224
Density	2.1×10^{-3}	4.0×10^{-4}	1.2×10^{-3}	1.4×10^{-2}	5.2×10^{-4}

A. Evaluation protocols

For each dataset, it is divided into training, validation, and testing sets at a ratio of 7:2:1. Use Recall@N and NDCG@N as evaluation metrics. Recall@N is the ratio of the number of real interactive items in the top N items in the recommended list to the total number of real interactive items, which is used to measure the coverage of the predicted results.

$$Recall@N = \frac{TP}{(TP + FN)} \tag{19}$$

Where TP represents that the true class of the sample is positive, and the final predicted result is also positive. Where FN represents that the true class of the sample is positive, but the final predicted result is negative. NDCG@N further considers the positions of each item in the recommended list.

$$NDCG@N = \frac{DCG}{IDCG} \tag{20}$$

$$DCG = \sum_{i=1}^{N} \frac{rel_i}{log_2(i+1)} \tag{21}$$

$$IDCG = \sum_{i=1}^{|REL|} \frac{rel_i}{log_2(i+1)} \tag{22}$$

Where rel_i represents whether the item at the i-th position is a user's actual interaction. It is denoted as 1 for true and 0 for false. Where REL represents ranking the results in the optimal order.

B. Baselines

The following 12 baseline methods are selected for comparison with the proposed SHTAU framework.

- NCF [2]: Instead of using the inner product, this method employs a neural architecture capable of learning an arbitrary function from the provided data. The NeuMF variant is used here for comparison.
- AutoR [3]: This method uses a behavioral reconstruction task to train for better user/item representation.
- PinSage [14]: This method combines efficient random walks and graph convolutions to generate embeddings of nodes that incorporate both graph structure as well as node feature information.
- NGCF [15]: This graph convolution-based approach additionally takes source-target representation interaction learning into consideration when designing its graph encoder.
- STGCN [7]: This model combines a graph convolutional encoder and a graph autoencoder to enhance model robustness during cold start scenarios and sample sparseness.
- GCCF [23]: This method improves recommendation performance by eliminating non-linearities and introduces a residual network structure tailored for collaborative filtering with user-item interaction modeling. This structure effectively addresses the over-smoothing issue encountered in graph convolution aggregation operations when dealing with sparse user-item interaction data.
- LightGCN [8]: This method provides a deep analysis of the efficiency of standard GCN and a simplified GCN model for the recommendation.
- HyRec [19]: A sequence model using hypergraph structures to learn high-dimensional connections for items.
- DHCF [18]: This model is a dual-channel hypergraph collaborative filtering framework.
- MHCN [20]: This model uses a multi-channel hypergraph convolutional network to enhance social recommendation by leveraging high-order user relations.
- SLRec [24]: This model adds contrastive learning between node features as a regular term to enhance the existing recommender system.
- SGL [25]: The model leverages random walks and feature dropout techniques to generate multiple views. It augments LightGCN by incorporating self-supervised contrastive learning, thereby enhancing its performance.

C. Experimental parameters

For a fair comparison, set the parameters of all baselines to the best values in their original papers. For the framework proposed in this paper, the implementation uses TensorFlow and uses the Adam optimizer, with a learning rate of 1e-3

TABLE III
PERFORMANCE COMPARISON ON FIVE DATASETS

data	metric	NCF	AutoR	PinSage	NGCF	STGCN	LightGCN	GCCF	HyRec	DHCF	MHCN	SLRec	SGL	SHTAU	Improv.
yelp	Recall@20	0.0252	0.0259	0.0345	0.0294	0.0309	0.0482	0.0462	0.0472	0.0449	0.0503	0.0476	0.0526	0.0744	41.4%
	NDCG@20	0.0202	0.0210	0.0288	0.0243	0.0262	0.0409	0.0398	0.0395	0.0381	0.0424	0.0398	0.0444	0.0652	46.8%
	Recall@40	0.0371	0.0504	0.0585	0.0522	0.0504	0.0803	0.076	0.0774	0.0751	0.0826	0.0821	0.0869	0.1007	15.8%
	NDCG@40	0.0227	0.0301	0.0373	0.033	0.0332	0.0527	0.0508	0.0511	0.0493	0.0544	0.0541	0.0571	0.0658	15.2%
Gowalla	Recall@20	0.0171	0.0239	0.0576	0.0552	0.0369	0.0985	0.0951	0.0901	0.0931	0.0955	0.0925	0.1030	0.1324	28.5%
	NDCG@20	0.0106	0.0181	0.0373	0.0298	0.0217	0.0593	0.0535	0.0498	0.0505	0.0574	0.0581	0.0623	0.0786	26.1%
	Recall@40	0.0216	0.0343	0.0892	0.081	0.0542	0.1431	0.1392	0.1306	0.1356	0.1392	0.1305	0.1500	0.1731	15.4%
	NDCG@40	0.0118	0.016	0.0417	0.0367	0.0262	0.071	0.0684	0.0669	0.066	0.0689	0.068	0.0746	0.0874	17.1%
Tmall	Recall@20	0.0082	0.0103	0.0202	0.0180	0.0146	0.0225	0.0209	0.0233	0.0156	0.0203	0.0191	0.0268	0.0442	64.9%
	NDCG@20	0.0059	0.0072	0.0136	0.0123	0.0105	0.0154	0.0141	0.0160	0.0108	0.0139	0.0133	0.0183	0.0301	63.4%
	Recall@40	0.014	0.0174	0.0345	0.031	0.0245	0.0378	0.0356	0.035	0.0261	0.034	0.0301	0.0446	0.0628	40.8%
	NDCG@40	0.0079	0.0097	0.0186	0.0168	0.014	0.0208	0.0196	0.0199	0.0145	0.0188	0.0171	0.0246	0.0437	77.6%
Amazon	Recall@20	0.0093	0.0131	0.0103	0.0222	0.0192	0.0319	0.0317	0.0302	0.0280	0.0296	0.0285	0.0327	0.0547	67.2%
	NDCG@20	0.0049	0.0099	0.0158	0.0160	0.0144	0.0236	0.0243	0.0225	0.0202	0.0219	0.0268	0.0249	0.0407	63.4%
	Recall@40	0.0223	0.0202	0.0366	0.0376	0.0312	0.0499	0.0483	0.0432	0.0471	0.0489	0.0463	0.0531	0.0834	57.0%
	NDCG@40	0.0133	0.0123	0.0124	0.021	0.0184	0.029	0.0285	0.0246	0.0272	0.0284	0.0314	0.0312	0.0494	58.3%
Mlens	Recall@20	0.0878	0.1230	0.1706	0.1611	0.1298	0.1789	0.1742	0.1801	0.1363	0.1497	0.1758	0.1833	0.2490	35.8%
	NDCG@20	0.1197	0.1667	0.2108	0.1961	0.1639	0.2128	0.2109	0.2178	0.1726	0.1814	0.2003	0.2205	0.2914	32.1%
	Recall@40	0.1634	0.1908	0.2724	0.2594	0.2006	0.265	0.2606	0.2685	0.2171	0.225	0.2633	0.2768	0.3081	11.3%
	NDCG@40	0.1427	0.1785	0.2362	0.2225	0.1782	0.2322	0.2331	0.234	0.1901	0.1962	0.236	0.2426	0.2853	17.6%

and a decay rate of 0.96.

D. Result analysis

Table 3 shows the experimental results.

From which the following conclusions can be drawn:

- The table indicates that the hypergraph neural network-based method outperforms other GNN-based baseline methods. This suggests that the incorporation of hypergraphs effectively captures high-order global connections and enhances recommendation performance.
- The SHTAU framework proposed in this paper consistently outperforms other baseline methods, showcasing its superiority. This can be attributed to several factors:

First, the SHTAU framework leverages the hypergraph transformer module to effectively capture and utilize global semantic connections.

Secondly, global-local self-Augmented learning extracts information from the hypergraph transformer module and guides the learning of local embeddings, thereby mitigating the influence of noise in the data.

Finally, the introduced AU module effectively optimizes the embeddings, achieving a good balance between Alignment and uniformity. The AU module enables users to be more evenly distributed in the latent space, thereby making each embedding express more information. Additionally, The AU module reduces the gap between embeddings of positive samples, making the relationship between users and items more explicit, eventually improving the overall performance.

E. Ablation experiments

To verify the rationality of the SHTAU framework structure, GCN, AU, and SSL modules were removed respectively to obtain 3 variants. Several variants were compared on three datasets, and the ablation experiment used Recall@N and NDCG@N as evaluation metrics; the experimental results are shown in Table 4. From the results, the following conclusions can be drawn:

- After removing the GCN module, the overall performance drops sharply. The results show that GCN embedding in the learning process is quite necessary.
- The removal of self-supervised learning (SSL) or AU modules significantly weakens the performance of SHTAU. At the same time, the positive effects of self-supervised learning and AU modules on performance are verified.

TABLE IV
ABLATION EXPERIMENT RESULTS

Variants	Yelp		Gowalla		Tmall	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
Off-GCN	0.0491	0.0413	0.0477	0.0275	0.0294	0.0198
Off-SSL	0.0701	0.0590	0.1276	0.0762	0.0430	0.0292
Off-AU	0.0651	0.0546	0.1232	0.0731	0.0387	0.0262
Full	0.0742	0.0652	0.1324	0.0786	0.0442	0.0301

F. Hyperparameter analysis

This section studies the impact of several key hyperparameters in our SHTAU framework.

a) The impact of hyperparameter λ_3

Fig 5 shows the performance comparison of different λ_3, which controls the scaling of $\mathcal{L}_{uniform}$. The lager the λ_3, the more scattered the embedding. The more evenly the embedding is distributed in the latent space, the more information can be expressed.

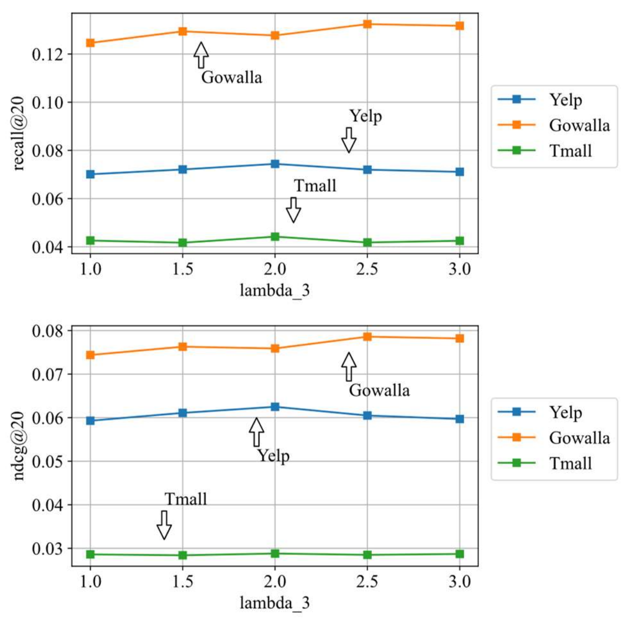


Fig.5 Performance comparison of different λ_3

From the experimental results, it can be observed that the optimal performance of the hyperparameter λ_3 varies for different datasets. The best performance is achieved with λ_3 is set to 2 on datasets yelp and tmall, while on dataset gowalla, the best performance is obtained with λ_3 is set to 2.5.

b) The impact of latent dimension

Fig 6 shows the performance comparison of different latent dimensions.

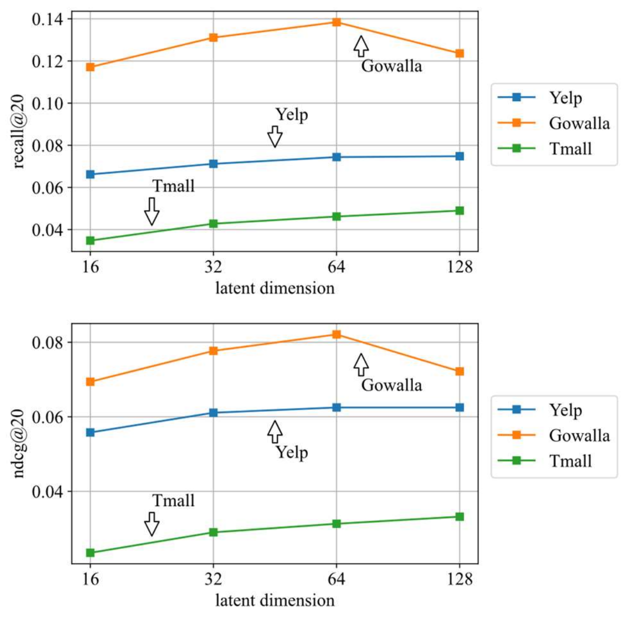


Fig.6 Performance comparison of different latent dimensions

Based on the experimental results, it can be observed that the optimal performance of the latent dimension varies across different datasets. Specifically, the best performance is achieved with a latent dimension of 128 on datasets such as Yelp and Tmall. This suggests that for these datasets, a higher dimensional space (128 dimensions) is better suited for capturing the underlying patterns and relationships in the data, resulting in improved recommendation performance. On the other hand, for the Gowalla dataset, the best performance is obtained with a lower latent dimension of 64. This indicates that for Gowalla, a lower-dimensional space is more effective in representing the user-item interactions and capturing the relevant information.

VI. Conclusion

This paper proposes a general self-supervised recommendation framework SHTAU, aiming to enhance robustness by self-reinforcing supervised signals, reduce the influence of data noise, and further improve performance by combining embedded aligned uniform auxiliary learning objectives. Extensive experiments are conducted, demonstrating the significant advantages over five real datasets compared to baseline methods.

In future work, we will investigate more target features of the embedded representation to improve recommendation performance. We will research how to construct auxiliary goals for multi-objective learning to improve recommendation performance.

References

[1] Yehuda Koren, Robert Bell, et al. 2009. Matrix factorization techniques for recommender systems. Computer 8 (2009), 30–37.

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In WWW. 173–182.FirstNameInitial.

[3] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In WWW. 111–112.

[4] 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.

[5] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, et al. 2019. Simplifying graph convolutional networks. In ICML. PMLR, 6861–6871.

[6] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In KDD

[7] Jiani Zhang, Xingjian Shi, Shenglin Zhao, et al. 2019. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. In IJCAI.

[8] Xiangnan He, Kuan Deng, Xiang Wang, et al. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In SIGIR. 639–648.

[9] Yifan Wang, Suyao Tang, et al. 2020. Disenhan: Disentangled heterogeneous graph attention network for recommendation. In CIKM. 1605–1614.

[10] 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).

[11] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, et al. 2021. Causal intervention for leveraging popularity bias in recommendation. In SIGIR. 11–20.

[12] Adit Krishnan, Ashish Sharma, et al. 2018. An adversarial approach to improve long-tail performance in neural collaborative filtering. In CIKM. 1491–1494.

[13] 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.

[14] Rex Ying, Ruining He, Kaifeng Chen, et al. 2018. Graph convolutional neural networks for web-scale recommender systems. In KDD. 974–983.

[15] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In SIGIR.

[16] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In SIGIR. 1001–1010.

[17] Yifan Wang, Suyao Tang, et al. 2020. Disenhan: Disentangled heterogeneous graph attention network for recommendation. In CIKM. 1605–1614.

[18] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In KDD. 2020–2029.

[19] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In SIGIR. 1101–1110.

[20] 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.

[21] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In International Conference on Machine Learning. PMLR, 9929–9939.

[22] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. arXiv preprint arXiv:2104.08821 (2021).

[23] 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.

[24] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, et al. 2021. Self-supervised Learning for Large-scale Item Recommendations. In CIKM. 4321–4330.

[25] 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.Diederik P. Kingma, Jimmy Lei Ba. Adam: A method for stochastic optimization[C]. //International Conference on Learning Representations. 2015.

[26] Ziang Li, Jie Wu, Guojing Han, Chi Ma, and Yuenai Chen, "Multi-hypergraph Neural Network with Fusion of Location Information for Session-based Recommendation," IAENG International Journal of Applied Mathematics, vol. 53, no.4, pp1389-1398, 2023