Dual-Attention Fusion Self-Supervised Graph Transformer for Recommendation

Shuhui Han, Dan Yang, Xi Gong

Abstract—**Recently, attention mechanisms have become increasingly prominent in recommendation algorithms, particularly for enhancing personalization and addressing computational complexity in large-scale datasets. However, traditional attention mechanisms frequently encounter obstacles, including high data sparsity and inadequate capture of domain information. This paper proposes a recommendation algorithm based on a Dual-Attention Fusion Self-Supervised Graph Transformer (DSGRec) to address these challenges. The model integrates sparse attention mechanisms with domain-aware attention mechanisms, thereby reducing computational complexity and enhancing the capture of domain-specific information. Furthermore, self-supervised learning bolsters the model's robustness and generalization capabilities. DSGRec leverages graph neural networks (GNNs) to capture global and local graph structural information, emphasizing key relationships through sparse processing while incorporating domain-aware feature mechanisms for dynamic adjustments of the attention weights across different domains. Experimental results on three real-world recommendation datasets demonstrated that DSGRec outperformed existing baseline models, particularly in sparse data scenarios, showing higher accuracy and more personalized recommendation capabilities.**

Index Terms—**Sparse Attention, Domain-aware Attention, Self-supervised Learning, Recommendation**

I. Introduction

With the rapid development of internet technologies and the continuous surge in data volume, as the demand for personalized services and content recommendations continues to grow, recommendation algorithms have become increasingly crucial. These algorithms are pivotal in enhancing user experience and commercial promotion by analyzing user behavior data and predicting preferences, as evidenced by the growing reliance on such technologies in sectors like e-commerce, news, and social media. However, despite the success of traditional recommendation algorithms (such as collaborative filtering[1] and content-based recommendation), they still face many challenges, especially in large-scale, sparse data scenarios with diverse user needs, where the accuracy and reliability of recommendation results remain limited. This paper investigates self-supervised[2] recommendation algorithms, aiming to accurately predict user needs and provide recommendations that meet these needs in large-scale and diverse contexts.

Recently, Graph Neural Networks (GNNs) and Transformer models have emerged, offering fresh insights and methodologies for enhancing recommendation algorithms. GNNs excel at modeling complex relationships between users and items, while Transformers[3] demonstrate outstanding capabilities in handling long-range dependencies and information fusion through their powerful attention mechanisms, particularly in tasks involving sequence modeling[4] and contextual understanding. Self-supervised learning, a training approach that does not require large amounts of labeled data, has also witnessed significant progress in recommendation algorithms. Predicting user behavior and item features, as demonstrated by the success of collaborative filtering and content-based filtering algorithms, significantly enhances the model's generalizability and robustness. However, despite the advancements in recommendation algorithms, challenges such as computational efficiency and information loss persist. For example, computational efficiency and information loss are issues. Traditional attention mechanisms[5] often suffer from high computational complexity and may fail to capture fine-grained information in sparse data. The challenge of domain awareness in recommendation algorithms is significant, as they must navigate the complexities of various domains or contexts to provide appropriate recommendations. However, existing attention mechanisms do not fully account for domain feature differences. Improving the utilization of graph-structured information remains an area of focus. The fusion of GNNs and Transformer models for graph data processing still has significant room for enhancement, particularly in recommendation tasks involving large-scale user behavior data and item relationship networks. The standard attention mechanism used in related work[6] has not captured the complex, fine-grained relationships between users and items. In particular, the attention matrix's computational demands can hinder models' deployment within large-scale recommendation systems, especially when dealing with sparse data. Furthermore, Although Transformer models excel at capturing long-range dependencies through global attention, in graph data processing applications, they often neglect domain-specific differences, such as user behavior and item features, resulting in limited domain awareness.

This paper proposes a recommendation algorithm based on the Dual-Attention Fusion Self-Supervised Graph Transformer (DSGRec) to address these issues. This approach aims to optimize the learning process of

Manuscript received January 2, 2025; revised June 5, 2025.

Shuhui Han is a postgraduate student at School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China (e-mail: hsh_yeying@163.com).

Dan Yang is a professor at School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China (corresponding author to provide email: asyangdan@163.com).

Xi Gong is a lecturer at School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China (e-mail: askdjy05gx@163.com).

graph-structured information by combining sparse attention[7] and domain-aware attention[8] mechanisms and introduces self-supervised learning further to enhance the accuracy and robustness of the model. The main contributions of this paper are summarized as follows:

- Combining sparse attention and domain-aware attention mechanisms to effectively handle large-scale sparse data while improving the model's personalized recommendation ability.
- Leveraging SSL-enhanced strategies and incorporating adaptive masking and adaptive enhancement mechanisms can significantly improve the performance and robustness of recommendation systems.
- Experimental validation shows that the proposed model outperforms existing methods across multiple recommendation tasks, especially in sparse data scenarios, where it significantly enhances both recommendation accuracy and personalization.

II. Related Work

This section will review some important research directions in the field of recommendation algorithms, with a focus on the combination of Graph Neural Networks (GNNs) and Graph Transformers[9], as well as the application of self-supervised learning in recommendation algorithms.

A. Recommendation Algorithms Based on Graph Transformer

With the rapid development of the recommendation system field, graph neural networks (GNNs) have emerged as a pivotal technology for addressing recommendation tasks. The prevalent utilization of graph data within recommendation systems allows GNNs to capture user-item interactions via the graph's structural connections, enhancing the precision of recommendation outcomes. Building on GNNs, Transformer models, known for their powerful self-attention[10] mechanisms, are introduced to recommendation systems to enhance the model's expressiveness and computational efficiency.

Graph Transformer models combine graph structure and self-attention mechanisms, constructing dependencies between nodes through the adjacency relationships of the graph to perform node embedding learning. Related work[11] proposes a new framework based on a continuous-time bipartite graph and a Time-Aware Collaborative Transformer (TCT) layer. This layer enhances the self-attention capability through a collaborative attention mechanism, allowing it to simultaneously capture the collaborative signals between users and items while considering the impact of temporal dynamics on sequential patterns. In related work[12], multi-path transformers extract aligned multimodal features from raw data and applied to top-k recommendation tasks. The Graph Transformer significantly improves performance when combined with the commonly used recommendation loss functions in the UGT model. In related work[13], a linear attention module is integrated with the Graph Transformer architecture to efficiently denoise noisy user/item embeddings.

B. Recommendation Algorithms Based on Self-Supervised Learning

Self-supervised learning[14] is a learning method that does not require a large amount of manually labeled data. In recent years, it has gained widespread attention in recommendation algorithms. Self-supervised learning harnesses unlabeled data by creating proxy tasks, allowing models to learn feature representations autonomously. By designing suitable proxy tasks, recommendation algorithms can autonomously discover latent information and user preferences from behavior data without needing labeled data, thus enhancing their performance and improving the model's recommendation accuracy and robustness. Related work[15] classifies existing self-supervised recommendation methods into four major categories: contrastive learning, generative learning, predictive learning, and hybrid learning, and introduces three key data augmentation methods that play an important role in self-supervised recommendation. Related work[16] proposes a multi-task self-supervised learning framework for large-scale item recommendations to address the label sparsity problem by capturing the latent relationships between item features. Related work[17] introduces a novel multimodal self-supervised learning method, which designs a data augmentation paradigm through adversarial perturbation, effectively capturing the intertwined effects of user interactions. Related work[18] presents a contrastive self-supervised learning framework for the sequential recommendation, introducing two information enhancement operators that generate high-quality views by leveraging the correlations between items. These correlations are subsequently utilized for contrastive learning.

C. Recommendation Algorithms Based on Self-Supervised Learning

Recently, the attention mechanism has gained widespread use in recommendation algorithms, mainly to improve their modeling of complex user-item relationships. The MIND model[19] is an attention-based model that employs two different attention mechanisms: the first is the self-attention mechanism[20], which allows the model to focus on different parts of the input sequence with varying degrees of importance, and the second is the additive attention mechanism[21], which computes a weighted sum of values based on the alignment scores. The MIND model calculates alignment scores using a feed-forward network with hidden layers and calculates the final scores by applying the softmax function to the alignment scores, which helps normalize the scores into a probability distribution. The NRMS[22] model introduces a neural news retrieval method with multi-head self-attention. The multi-head self-attention mechanism enhances news representation by learning from headlines, simultaneously modeling word interactions, and capturing the context of the news content more effectively.

III. Preliminaries

This section of the paper is an essential guide to employing fundamental symbols consistently throughout the document. It thoroughly explains each symbol's meaning, making it clear and understandable for readers. Furthermore, it concisely overviews the paper's main problem area. To further assist the reader, Table I has been included below.

This table provides a comprehensive list of specific symbol descriptions, allowing for quick reference and aiding in comprehending the complex concepts discussed within the paper.

TABLE I SYMBOL DESCRIPTION

Symbol	Description
$U=\{u_1,u_2,...,u_I\}$	the set of users
$I=\{i_1,i_2,...,i_J\}$	the set of items
$A\in\mathbb{R}^{I\times J}$	the interaction between the user and the item
$a_{i,j}$	the interaction between the user u_i and the item i_j
$G=\{V,E\}$	user-item interaction graph
$V=U\cup I$	the set of nodes of the graph, including all users and items
$E=\{e_{i,j}\mid a_{i,j}=1\}$	the set of edges in G
$\hat{y}_{i,j}$	the interaction score between the user u_i and the item i_j
$f(G;\Theta)$	function based on graph G and model parameters Θ

Definition 1. The collection of users and items. In a specific scenario where we have I number of users and J number of items, the set of users $U=\{u_1,u_2,...,u_I\}$ symbolizes the entire group of users. In contrast, the set of items $I=\{i_1,i_2,...,i_J\}$ represents the entire collection of items. This means that every user in this system is included within the set of users. Similarly, every item available or considered within this framework is included within the set of items.

Definition 2. Matrix of interactions between users and items. The matrix $A\in\mathbb{R}^{I\times J}$ represents the user-item interaction matrix, which is a matrix with I rows and J columns that $a_{i,j}$ indicate whether the user u_i interacts with the item i_j or not.

Definition 3. User-Item Interaction graph. Define a user-item interaction graph $G=\{V,E\}$, where V is the set of graph nodes representing all users and items in the system. The node set V includes all users and items, i.e. $V=U\cup I$. E is the set of edges of the graph, representing the interactions between users and items. The edge set E includes all user-item interaction pairs, i.e., indicating an interaction between the user u_i and the item i_j.

In DSGRec, the user-item graph G is constructed by transforming the interaction matrix from the original data. The interaction information $a_{i,j}=1$ indicates an interaction between the user u_i and the item i_j. Suppose $a_{i,j}=0$ it indicates no interaction between the user u_i and the item i_j. When the graph $G=\{V,E\}$ is constructed, the recommendation algorithm can predict the interaction between the user and the item through the graph's structure. The prediction function of the recommendation algorithm is defined as follows:

$$\hat{y}_{i,j}=f(G;\Theta) \tag{1}$$

Here, $\hat{y}_{i,j}$ it represents the predicted rating between the user and the item. At the same time, it is a function based on graph G and model parameters, which calculates the predicted value using graph structure and node features.

IV. ALGORITHM FRAMEWORK

This section provides a detailed introduction to the proposed recommendation algorithm, DSGRec, which is divided into four parts: 1) Feature Embedding, via anchor nodes, captures global topological information, refining user and item node embeddings. This enhances collaborative modeling and enriches representations for self-supervised learning; 2) Feature Extraction involves personalized knowledge transfer, deriving the final embedding representations of users and items by considering factors from both the user and item sides; 3) Adaptive Fusion leverages a graph masking autoencoder to enhance the user-item interaction graph, with a masking strategy that prioritizes relevant interactions, thereby improving the recommendation algorithm's performance; 4) Recommendation Prediction, which uses a loss-based approach combined with Bayesian Personalized Ranking (BPR) loss[23] for model optimization and the final recommendation prediction. The model diagram of DSGRec is shown in Figure 1.

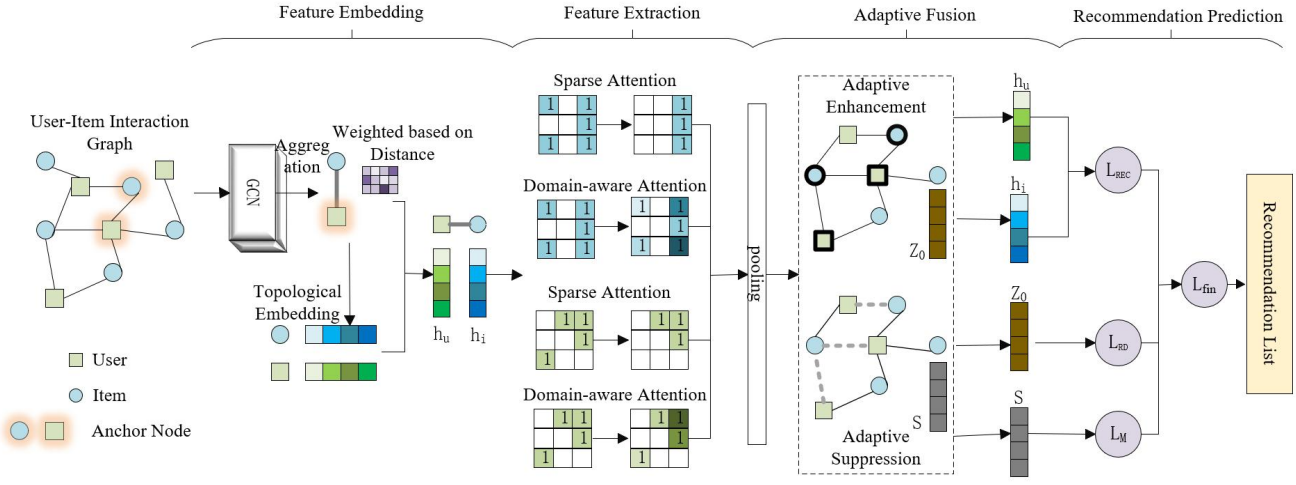

Fig. 1. Overall Framework of the DSGRec. The selected anchor nodes are aggregated through the GCN layer to obtain the aggregated topological embedding, with distances and weights labeled. After passing through sparse attention and domain-aware attention, a pooling operation is performed, followed by adaptive enhancement to obtain the final embedding.

A. Feature Embedding

First, several user-item pairs are selected from the user-item interaction graph $G=\{V,E\}$, referred to as the anchor node-set $V_A \subset V$, and sampled. Anchor nodes are crucial for capturing global topological information and enhancing the model's understanding of graph structure through their relationships with target nodes. Next, the distance between the target node and each anchor node is computed. For each target node v_k, the distance $d_{k,a}$ to each anchor node v_a is calculated. This distance is the minimum number of edges that must be traversed in the graph to reach from v_k to v_a. These distances measure topological proximity between the target and anchor nodes, forming the basis for subsequent embedding updates. The relevance weight $\omega_{k,a}$ between the target node v_k and the anchor node v_a is determined by their distance, and the calculation formula is as follows:

$$\omega_{k,a}=\begin{cases}\dfrac{1}{d_{k,a}+1}, & if d_{k,a}\le q\\ 0, & otherwise\end{cases} \tag{2}$$

Here, q is a maximum distance threshold that limits the influence of distant anchor nodes on the target node. In this way, the model can focus more on anchor nodes with a stronger association with the target node. During the propagation process in each graph layer, the target node embedding is updated based on the anchor node weights. The formula is as follows:

$$\tilde{h}_k^l=\sum_{v_a\in V_A}\mathrm{W}^l\cdot\omega_{k,a}\cdot\left[\tilde{h}_k^{l-1}\middle\|\tilde{h}_a^{l-1}\right]/|V_A| \tag{3}$$

$\tilde{h}_k^l$ is the embedding representation of the target node v_k at the l-1-th layer. $\tilde{h}_k^{l-1}$ And $\tilde{h}_a^{l-1}$ represent the embeddings of the target node v_k and anchor node v_a at the l-1-th layer, respectively. W^l is a learnable linear transformation matrix used to map node embeddings into a new feature space. $[\bullet\|\bullet]$ Denotes the concatenation operation of vectors from the target node embedding to the anchor node embedding. After multiple layers of graph propagation (a total of L layers), the resulting embedding matrix $\tilde{H}^L$ contains higher-order global topological information of the graph. These embeddings are further injected into their corresponding node representations to form the final global topological embeddings, as shown in the following formula:

$$\tilde{H}=TE\left(H;\left\{W_l^T\right\}\right) \tag{4}$$

TE is a parameterized Transformer designed to incorporate global topological information into node embeddings, capturing the complex collaborative relationships between user embedding h_u and items embedding h_i ($h_u,h_i\in\overline{H}$).

B. Feature Extraction

The attention layer of DSGRec is implemented by combining sparse attention and domain-aware attention. In graph neural networks, the attention layer is crucial for determining the attention weights that reflect node similarity. These weights are then leveraged to update the node embeddings, effectively capturing the graph's structure and relationships. These attention weights are based on the graph's adjacency information and the relationships between nodes, and they are ultimately used to improve the embedding representations of the recommendation model.

1) Sparse Attention

In DSGRec, the sparse attention module calculates attention weights only between adjacent nodes in the user-item graph to reduce computational complexity while preserving local relationships between nodes. Sparse attention aims to highlight key node relationships efficiently by minimizing computations on non-essential pairs. For any given node pair (v_k,v_a), sparse attention selects important connections by calculating the weighted attention weights $s_{k,a}$ with the specific computation formula as follows:

$$s_{k,a}=Soft\max\left(\frac{(h_kW_Q)\cdot(h_aW_k)^{\mathrm{T}}}{\sqrt{d}}\right) \tag{5}$$

Here h_a are the embeddings of the target node v_k and item v_a, respectively. W_Q and W_K are the learned query and key projection matrices, where d represents the dimension of the embeddings used for scaling purposes. The softmax function confines the range of the attention weights.

Then, through the sparsification operation, the attention range is restricted to the high-importance neighborhood:

$$\tilde{s}_{k,a}=\begin{cases}s_{k,a}, & if d_{k,a}\le q,\\ 0, & otherwise\end{cases} \tag{6}$$

$d_{k,a}$ represents the distance between the nodes, and q is the predefined maximum distance threshold. Sparse attention is computed only for adjacent nodes, avoiding the high computational complexity of calculating over the entire graph, and it effectively captures the local interaction information between users and items.

2) Domain-Aware Attention

To address the domain discrepancy problem arising from interactions between users and items, such as clicks, favorites, etc., domain-aware attention computes the similarity between nodes based on different interaction types and dynamically adjusts the attention weights. This improves the interaction between nodes of different interaction types in the graph. Domain-aware attention adjusts the weights of different domains by introducing domain embeddings, enhancing the ability to capture cross-domain semantics. The domain-aware similarity is calculated using the following formula:

$$g_{k,a}=\sigma(h_kW_Rh_a^{\mathrm{T}}+Domain(v_k,v_a)) \tag{7}$$

W_R is the learned matrix for the domain embeddings. $Domain(v_k,v_a)$ is the weighted function of domain relevance, typically computed by the dot product of the domain embeddings, and σ is the activation function.

Based on the domain-aware similarity, the domain attention weight $w_{k,a}$ is given by:

$$w_{k,a}=s_{k,a}\cdot g_{k,a} \tag{8}$$

This means that the sparse and domain-aware attention weights are multiplied for fusion.

3) Fusion of Dual Attention

Fusing these two attention mechanisms allows the model

to consider the relationships between nodes and dynamically adjust the weights based on the interaction types between users and items, thus achieving more detailed node representations. To integrate the efficiency of sparse attention with the semantic modeling capabilities of domain-aware attention, this paper introduces a fusion module that blends the outcomes of both attention mechanisms with weighted consideration. Using the overall performance of the model. The final attention weight is given by:

$$\alpha_{k,a} = w_{k,a} \cdot \frac{\exp((h_k W_Q)\cdot(h_a W_K)^{\mathrm{T}})}{\sum_{b\in N(k)} \exp((h_k W_Q)\cdot(h_b W_K)^{\mathrm{T}})} \tag{9}$$

The aggregation is performed to obtain the probability scores for the graph edges:

$$\tilde{\alpha}_{k,a} = \sum_{h=1}^{H} \alpha_{k,a}^{h} / H \tag{10}$$

$$p(v_k, v_a \mid G) = \frac{\tilde{\alpha}_{k,a}}{\sum_{(v_k,v_a)\in\varepsilon} \tilde{\alpha}_{k,a}} \tag{11}$$

From the edge set E, edges are independently sampled according to their probability scores $p(v_k, v_a \mid G)$, where $\rho_R \cdot |E|$ edges are selected. The hyperparameter $\rho_R \in \mathbb{R}$ controls the size of the selected important subset of edges.

C. Adaptive Fusion Module

This module provides personalized features for individual tasks, delivering customized suggestions catering to users' needs. More precisely, the model utilizes embeddings generated by the graph Transformer and applies advanced reasoning methods to predict user preferences across various items. This process can be formally represented as follows:

$$\bar{y}_{i,j} = z_i^{L\mathrm{T}} \cdot z_j^L; z_k^L = \sum_{(v_k,v_a)\in\varepsilon_R} \beta_{k,a} \cdot z_a^{l-1} \tag{12}$$

$$Z^0 = GT(G, TE(H)) = \|_{h=1}^{H} \sum_a \alpha_{k,a}^h W_V^h \bar{h}_a + \bar{h}_k \tag{13}$$

$\bar{y}_{i,j} \in \mathbb{R}$ represents the predicted probability that the user u_i will choose item i_j . The embeddings z_i^L and $z_j^L \in \mathbb{R}^d$ are used to predict the interaction between the user and item, while the embedding z_k^L of vertex v_k is obtained through L layers of LightGCN. E_R represents the sampled edge set from the graph G_R .In $\beta_{k,a} = 1/\sqrt{d_k d_a}$, d_k and d_a are the degree-normalized Laplacians of nodes v_k and v_a . The zero-order embeddings Z^0 are aggregated through the multi-head attention of the topology-aware graph Transformer, where H is the number of attention heads and $W_V^h \in \mathbb{R}^{d/H\times d}$ represents the value transformation matrix in the self-attention mechanism. A residual connection is employed, with the topology-aware embeddings $\bar{h}_a$ serving as input.

1) Adaptive Masking

The adaptive module first computes the relevance score for each user-item interaction edge to determine which edges represent the most important interaction relationships in the recommendation algorithm. These scores generate the masked graph $G_M = \{V, E_M\}$, where edges with high relevance scores are retained, while edges with low scores are masked. The generation of the masked graph is based on sampling the inverse of the relevance scores:

$$E_M \sim p_M(E_M \mid G) = \prod_{(v_k v_a)\in E_M} \alpha_{k,a}^M \prod_{(v_k v_a)\in E\setminus E_M} \alpha_{k,a}^M \tag{14}$$

$$|E_M| = \rho_M |E|; \alpha_{k,a}^M = \frac{\bar{\alpha}_{k,a}^M}{\sum_{(v_k,v_a)\in E} \bar{\alpha}_{k,a}^M}; \bar{\alpha}_{k,a}^M = \frac{1}{\bar{\alpha}_{k,a} + \varepsilon} \tag{15}$$

In the masked graph, the edge sampling probability is denoted as $\rho_M(\cdot)$, $\alpha_{k,a}^M$ representing the probability of selecting the edge between nodes v_k and v_a in the masking generator. $\bar{\alpha}_{k,a}^M$ is the unnormalized attention score, which is computed from the inverse of the edge weight $\alpha_{k,a}$, and a small value ε is added to avoid division by zero. The edge density in the masked graph is higher than that of the reasoning graph, ensuring that only the most important reasoning edges are removed, thus achieving noise-resistant encoding. The masked graph G_M with edge set E_M is input to the autoencoder network, and the process is represented as follows:

$$S = GT(G_M, TE(\bar{S}^L)); \bar{s}^l = \sum_{(v_k,v_a)\in E_M} \beta_{k,a} \cdot \bar{s}_a^{l_1} \tag{16}$$

where $S \in \mathbb{R}^{(I+J)\times d}$ represents the final embedding in the autoencoder. $GT(\cdot)$ and $TE(\cdot)$ represent the graph Transformer network and the topology information encoder, respectively. The initial embeddings of the L-layer local node $\bar{S}^L$ encoded by LightGCN are the initial embeddings. The edge density in the masked graph is higher than in the relevance graph, meaning that only the most important relevance edges are removed in the masked graph. This reduces the noise impact and enhances the model's stability.

2) Adaptive Enhancement

The generated masked graph G_M is input into the graph masking autoencoder, which learns valuable interaction patterns by reconstructing the masked user-item interactions. Specifically, the graph masking autoencoder workflow involves the following steps: The graph transformer and topology information encoder process the masked graph. The graph transformer integrates the masked node embeddings with locally encoded node embeddings (via LightGCN) to refine the initial node representations. Reconstruction Objective: The autoencoder aims to reconstruct the masked user-item interactions. The model learns important interaction patterns during training by minimizing the reconstruction error.

Adaptive enhancement enhances the user-item interaction graph via graph masking autoencoder, utilizing relevance score-based masking. It enables the model to concentrate on crucial interaction relationships, thus significantly boosting recommendation algorithm performance. Adaptive enhancement minimizes noise impact, yielding more precise feature representations for SSL.

D. Recommendation List Generation

The model determines user preferences for items by calculating user and item embeddings. Specifically, its prediction task revolves around estimating the interaction scores between users and items. Various loss functions are devised to enhance the model's predictive accuracy, each playing a pivotal role in distinct modules, thereby aiding the model in discovering the underlying patterns of user-item

interactions.

For each user u_i and item i_j, their interaction score is calculated as follows:

$$\hat{y}_{i,j} = s_i^{\mathrm{T}} s_j \tag{17}$$

s_i and s_j are the embedding representations of the user u_i and item i_j. These embeddings are obtained through the propagation process of the graph transformer, capturing the complex interactions between users and items.

During the training phase, the embedding matrix $S \in \mathbb{R}^{(I+J)\times d}$ is used to train the recommendation model. The goal of model optimization is to minimize the following pointwise loss function:

$$L_{\mathrm{Rec}} = \sum_{a_{i,j}=1} -\log \frac{\exp(s_i^{\mathrm{T}} s_j)}{\sum_{p' \in P} \exp(s_i^{\mathrm{T}} s_j)} \tag{18}$$

The L_{MAE} loss is used to reconstruct the masked user-item interaction patterns, and the objective of the adaptive enhancement module is to learn more effective interaction representations by minimizing the reconstruction error. The specific calculation is as follows:

$$L_{MAE} = \sum_{(v_k, v_a) \in E \setminus E_M} (-\hat{y}_{k,a}) \tag{19}$$

By masking the important edges in the graph structure and using the graph transformer for reconstruction, the loss function L_{MAE} helps the model focus on more important interaction patterns, reducing noise interference.

The model predicts user preferences for items based on the embeddings and reasoning information extracted by the graph transformer, with the calculation process as follows:

$$\bar{y}_{i,j} = z_i^L \cdot z_j^L, z_k^L = \sum_{(v_k, v_k') \in E_R} \beta_{k,k'} \cdot z_k^{L-1} \tag{20}$$

$\bar{y}_{i,j}$ represents the predicted score of the user u_i for the item i_j, and z_k^L is the L-th layer embedding of vertex v_k. The aggregation is performed using the edge set of the sampled reasoning graph E_R, which $\beta_{k,k'} = 1/\sqrt{d_k \cdot d_{k'}}$ is the Laplacian normalization coefficient based on the vertex degrees. The model completes multi-head embedding aggregation through the topology-aware graph Transformer, combining residual connections to obtain the zero-order embedding Z_0. Based on the predicted score for each user-item interaction (u_i, i_j), the calculation is as follows:

$$L_{\mathrm{RD}} = \sum_{(u_i, p_j^+, p_j^-)} -\log \sigma(\bar{y}_{i,j^+} - \bar{y}_{i,j^-}) \tag{21}$$

(u_i, p_j^+, p_j^-) forms a triplet of user, positive, and negative items, and $\sigma(\cdot)$ is the sigmoid function.

By combining all of the aforementioned loss functions, the model completes training by optimizing the following objective function:

$$L = L_{\mathrm{Rec}} + L_{MAE} + \lambda_1 L_{RD} + \lambda_2 \cdot \|\Theta\|_F^2 \tag{22}$$

λ_1 and λ_2 are hyperparameters used to balance the loss functions. The last term is the Frobenius norm regularization applied to the parameters to prevent overfitting.

V. Experiments

This section introduces the datasets and evaluation metrics used in the experiments, presents extensive comparative and ablation studies, and provides a detailed analysis of the experimental results.

A. Experimental Setup

1) Datasets

The model is tested on three widely used real-world datasets to evaluate recommendation algorithms. These datasets include Yelp, Ifashion, and LastFM, covering user rating behaviors across various items. The Yelp dataset contains user reviews of businesses and is widely used for location-based research and business recommendations. iFashion is a publicly accessible dataset focused on fashion-related research, providing suggestions for appropriate fashion items.is based on user preferences and historical behaviors. The LastFM dataset, known for its extensive user ratings and listening history, is a cornerstone in music recommendation research, as evidenced by its application in studies such as those in references 1 and 2. Detailed information is provided in Table II.

TABLE II Statistical Information Of The Datasets

Datasets	Yelp	Ifashion	LastFM
Users	42,712	31,668	1,889
Items	26,822	38,048	15,376
Interaction	182,357	618,29	51,987
Density	$1.6\,e^{-4}$	$5.1\,e^{-4}$	$1.8\,e^{-3}$

2) Evaluation Metrics

Two common evaluation metrics are used to assess the recommendation performance of the model: Recall@k and Normalized Discounted Cumulative Gain (NDCG@k), where @k indicates the ranking position, typically referring to the top-k items in the recommendation list. In this case, k=10,20,40.

- Recall@k. This measures the overlap between the items the algorithm recommends and the items the user is interested in. It is commonly used to evaluate the comprehensiveness of the recommendations. The calculation is as follows:

$$\mathrm{Re}call@k = \frac{\left| \left\{ i \in \tau_u^{rel} : i \in \hat{\tau}_u^{(k)} \right\} \right|}{\left| \tau_u^{rel} \right|} \tag{23}$$

  Where τ_u^{rel} represents the set of all relevant items for user u, and $\hat{\tau}_u^{(k)}$ are the top-k items recommended by the algorithm for user u.

- NDCG@k. This is a metric used to evaluate the ranking quality. It considers not only the relevance of the recommended items but also the order in which they are presented. NDCG is particularly suitable for ranking-based recommendation tasks (such as movies, products, etc.), and the calculation is as follows:

$$NDCG@k = \frac{1}{k} \sum_{j=1}^{k} \frac{1}{\log_2(i_j + 1)} \tag{24}$$

  Where i_j denotes the position of the relevant j-th item in the recommendation list. If the j-th item is not in the recommendation list, the $\frac{1}{\log_2(i_j+1)}$ becomes 0.

To evaluate the model's performance, metrics such as precision and recall are used to assess the top-10, top-20, and top-40 items in the recommendation list. The interaction data of each dataset is split into training, validation, and test sets in 0.7:0.05:0.25 ratio. A comprehensive ranking protocol is employed to evaluate the accuracy of recommendations for users across the entire item set. This protocol helps reduce evaluation bias caused by negative sampling. The higher the metric value, the better the recommendation performance of the DSGRec model.

3) Baselines

The DSGRec model is compared with the following 11 baseline methods:

- BiasMF[24]. Through matrix factorization, BiasMF maps users and items into a latent space and introduces bias terms to account for the personalized preferences of users and items, thereby improving prediction accuracy.
- NGCF[25]. NGCF captures higher-order collaborative information through a multi-layer graph neural network, thus delving into the complex relationships between users and items.
- AutoRec[26]. AutoRec uses an autoencoder structure to learn user and item embedding representations by reconstructing observed interactions, making it particularly suitable for handling sparse data.
- GCCF[27]. GCCF leverages graph convolutional networks (GCNs) to capture higher-order information on users and items for collaborative filtering, removing nonlinear transformations and introducing residual connections to enhance the stability of graph convolutions and recommendation accuracy.
- LightGCN[28]. LightGCN is a simplified version of graph convolutional networks, commonly used for recommendation tasks, which remove unnecessary feature transformations and nonlinear activations.
- NCL[29]. NCL uses the EM algorithm to cluster users and then performs neighborhood-enhanced contrastive learning within the clusters to improve recommendation accuracy.
- HCCF[30]. NCCF builds global and local views based on hypergraphs for contrastive learning, better capturing the complex relationships between users and items.
- SGL[31]. SGL builds multiple views through random data augmentation techniques (such as node dropout, edge dropout, etc.), enhancing the model's learning of interaction structures and improving robustness.
- GFormer[32]. GFormer combines graph neural networks and Transformer architecture in a rationally aware generative self-supervised learning process, focusing on capturing long-range dependencies in graph-structured data.
- PinSage[33]. PinSage combines graph convolutional networks with a random walk-based message-passing mechanism to efficiently encode large-scale user-item interaction graphs.
- SLRec[34]. This method employs contrastive learning on node features to enhance the regularization of recommendation learning.

TABLE III PERFORMANCE COMPARISON OF DSGREC WITH BASELINES ON THREE DATASETS

Datasets	Metric	BiasMF	AutoRec	PinSage	NGCF	GCCF	LightGCN	SLRec	NCL	HCCF	SGL	GFormer	DSGRec
LastFM	Recall@10	0.0609	0.0543	0.0899	0.1257	0.1230	0.1490	0.1133	0.1491	0.1502	0.1496	0.1574	**0.1591**
	NDCG@10	0.0696	0.0599	0.1046	0.1498	0.1452	0.1739	0.1384	0.1745	0.1773	0.1775	0.1831	**0.1858**
	Recall@20	0.0980	0.0887	0.1343	0.1918	0.1816	0.2188	0.1747	0.2196	0.2210	0.2256	0.2355	**0.2397**
	NDCG@20	0.0860	0.0769	0.1229	0.1759	0.1681	0.2018	0.1613	0.2021	0.2047	0.2070	0.2142	**0.2179**
	Recall@40	0.1450	0.1550	0.1990	0.2794	0.2649	0.3156	0.2533	0.3130	0.3180	0.3156	0.3300	**0.3331**
	NDCG@40	0.1067	0.1031	0.1515	0.2146	0.2049	0.2444	0.1930	0.2437	0.2484	0.2498	0.2567	**0.2591**
Yelp	Recall@10	0.0122	0.0230	0.0278	0.0438	0.0484	0.0422	0.0422	0.0493	0.0518	0.0522	0.0561	**0.0569**
	NDCG@10	0.0070	0.0133	0.0171	0.0269	0.0296	0.0254	0.0259	0.0301	0.0318	0.0319	0.0350	**0.0352**
	Recall@20	0.0198	0.0410	0.0454	0.0678	0.0754	0.0761	0.0650	0.080	0.0796	0.0819	0.0878	**0.0885**
	NDCG@20	0.0090	0.0186	0.0224	0.0340	0.0378	0.0371	0.0327	0.0402	0.0391	0.0410	0.0442	**0.0449**
	Recall@40	0.0303	0.0678	0.0712	0.1047	0.1163	0.1031	0.1026	0.1193	0.1244	0.1249	0.1328	**0.1345**
	NDCG@40	0.0117	0.0253	0.0287	0.0430	0.0475	0.0411	0.0418	0.0482	0.0510	0.0517	0.0551	**0.0558**
Ifashion	Recall@10	0.0302	0.0309	0.0291	0.0375	0.0373	0.0437	0.0373	0.0474	0.0489	0.0512	0.0542	**0.0547**
	NDCG@10	0.0281	0.0264	0.0276	0.0350	0.0352	0.0416	0.0353	0.0446	0.0464	0.0487	0.0520	**0.0523**
	Recall@20	0.0523	0.0537	0.0505	0.0636	0.0639	0.0751	0.0633	0.0797	0.0815	0.0845	0.0894	**0.0899**
	NDCG@20	0.0360	0.0351	0.0352	0.0442	0.0445	0.0528	0.0444	0.0558	0.0517	0.0603	0.0625	**0.0642**
	Recall@40	0.0858	0.0921	0.0851	0.1062	0.1047	0.1207	0.1043	0.1283	0.1308	0.1345	0.1425	**0.1432**
	NDCG@40	0.0474	0.0483	0.0470	0.0585	0.0584	0.0677	0.0583	0.0723	0.0744	0.0773	0.0818	**0.0823**

4) Parameter Setting

To ensure a fair comparison, the DSGRec model is implemented using the PyTorch framework, and the Adam optimizer is used for parameter updates, with a learning rate set to 1 ℓ^{-3} and no learning rate decay applied. For the hyperparameters of the DSGRec model, the embedding dimension is set to d=3 by default, the size of the anchor node-set is set to $|V_A| = 32$, and the graph inference retention rate ρ_R is adjusted within the range of [0.5, 0.9]. The coefficients for the various loss functions are searched and optimized. Within the following ranges $\lambda_1 \in \{0.5,1,2,4,8\}$, $\lambda_2 \in \{1,1e^{-1},1e^{-2},1e^{-3},1e^{-4}\}$, $\lambda_3 \in \{1e^{-3},1e^{-4},1e^{-5},1e^{-6},1e^{-7},1e^{-8}\}$. Additionally, the number of graph Transformer layers is adjusted within the range of [1, 6], and the number of graph convolutional layers is tuned within the range of [1, 5].

B. Analysis of Experimental Results

Extensive experiments are conducted on the LastFM, Yelp, and iFashion datasets, and compared with ten baseline methods. The experimental results are shown in Table III. The results indicated that:

- The models (BiasMF, AutoRec) utilize traditional collaborative filtering methods, where BiasMF optimizes matrix factorization by introducing bias scores, and AutoRec learns embedding representations by reconstructing interaction data using an autoencoder. These methods demonstrate effectiveness in recommendation tasks. The models (PinSage, NGCF, LightGCN, GCCF) are based on graph neural network methods, which effectively capture higher-order collaborative signals between users and items through different message-passing mechanisms. These models, namely NCL, SGL, and HCCF, significantly improve recommendation performance by leveraging self-supervised learning strategies. NCL integrates clustering with contrastive learning, SGL employs various data augmentation techniques to create views for contrastive learning, and HCCF constructs global and local views using hypergraphs to enhance performance. The introduction of self-supervised learning proves its significant effect on recommendation algorithms.
- Experimental results indicate that the DSGRec model consistently surpasses all other baseline models, achieving notable enhancements in evaluation metrics. This underscores the significant improvement in model performance attributed to the fusion of sparse attention and domain-aware attention within the attention layer. Among the datasets, the LastFM dataset stands out for its rich user-item interaction data. Users on this platform exhibit diverse interests, which could greatly benefit from the application of sparse attention and domain-aware attention.

C. Model Analysis

This section delves into the efficacy of each module within the DSGRec model, scrutinizing the influence of pivotal parameters. It is accompanied by pertinent ablation experiments and a detailed parameter sensitivity analysis to understand better their respective roles in the model's overall performance. We will explore how each component of the DSGRec model contributes to its effectiveness, examining the key parameters that drive its functionality. Through a series of ablation studies, we aim to isolate and evaluate the impact of individual modules, thereby gaining insights into their importance. Additionally, a comprehensive parameter sensitivity analysis will be conducted to assess how variations in these parameters affect the model's behavior and accuracy. This thorough examination will provide a deeper understanding of the intricate interplay between the model's components and performance, ultimately shedding light on optimizing the DSGRec model for enhanced outcomes.

1) Ablation Experiments

In order to ascertain the extent to which data augmentation techniques and attention mechanisms contribute to the model's performance, two distinct variants of the DSGRec model are created by systematically removing various modules. To be precise, DSGRec-n stands for a variant of the model that does not employ sparse attention when computing the relationships between adjacent nodes. At the same time, DSGRec-a denotes a variant that does not utilize domain-aware attention to fine-tune the attention weights.

TABLE IV PERFORMANCE COMPARISON OF DSGREC WITH OTHER ABLATION METHODS

Datasets	Metrics	DSGRec-n	DSGRec-a	DSGRec
Yelp	Recall@10	0.0563	**0.0571**	0.0569
	NDCG@10	0.0351	**0.0353**	0.0352
	Recall@20	**0.0888**	0.0878	0.0885
	NDCG@20	0.0446	0.0439	**0.0449**
	Recall@40	0.1341	0.1339	**0.1345**
	NDCG@40	0.0553	0.0554	**0.0558**
LastFM	Recall@10	0.1579	0.1580	**0.1591**
	NDCG@10	0.1844	0.1839	**0.1858**
	Recall@20	0.2362	0.2366	**0.2397**
	NDCG@20	0.2153	0.2149	**0.2179**
	Recall@40	0.3305	0.3308	**0.3317**
	NDCG@40	0.2571	0.2570	**0.2577**
Ifashion	Recall@10	0.0544	0.0547	**0.0547**
	NDCG@10	**0.0525**	0.0520	0.0523
	Recall@20	0.0897	0.0898	**0.0899**
	NDCG@20	0.0640	**0.0647**	0.0642
	Recall@40	0.1429	**0.1432**	**0.1432**
	NDCG@40	0.0821	0.0819	**0.0823**

To illustrate the impact of these changes, experiments are conducted using the Yelp, the Ifashion, and LastFM datasets, with the results presented in Table IV.

To evaluate the contributions of the key components in DSGRec, we conduct ablation experiments by removing the sparse attention mechanism and the domain-aware attention mechanism separately. As shown in Table IV, the complete DSGRec consistently achieves the best performance across all three datasets and most evaluation metrics, indicating the

effectiveness of the proposed architectural components. Specifically, the DSGRec-n variant, which omits the sparse attention mechanism, shows a slight but consistent drop in performance. For instance, Recall@40 decreases from 0.1345 to 0.1341 on the Yelp dataset, and NDCG@20 decreases from 0.0449 to 0.0446. This suggests that the sparse attention mechanism is crucial in filtering noisy or less relevant neighboring information, enabling more focused and informative user-item interaction modeling.

Similarly, the DSGRec-a variant, which removes the domain-aware attention module, performs worse than the entire model, particularly on the LastFM and iFashion datasets. The decline in NDCG scores (e.g., from 0.2179 to 0.2149 on LastFM at NDCG@20) highlights the importance of domain-aware attention in adjusting attention weights according to domain-specific characteristics. This adaptive weighting strategy helps the model capture heterogeneous semantics across different domains better. Overall, these ablation results demonstrate that both modules—sparse attention and domain-aware attention—complement each other in enhancing the expressiveness and precision of the learned representations, contributing significantly to the overall performance of DSGRec.

2) Analysis of Hyperparameters

Discussion on the Impact of Hyperparameters b2-value and keeprate on DSGRec Results.

- Impact of b2-value. b2-value controls the influence of the BPR loss. When b2's value becomes excessively high, the BPR loss may dominate the overall loss, changing the model's optimization path. Using the LastFM dataset as an example, with all other hyperparameters fixed, the b2-value is adjusted between 1 and 4. The experimental results are shown in Figure 2:

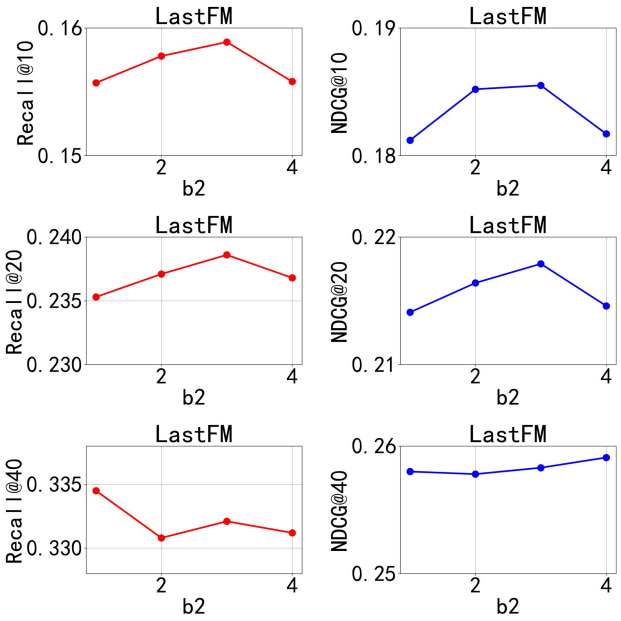

Fig. 2. Performance Comparison of different b2-values

The figure shows that the model's performance peaks when b2=3. This suggests that the BPR loss function is pivotal in the recommendation model's loss function.

- Impact of keeprate

keeprate controls the proportion of edges retained when constructing the graph's relevance. Extracting a subgraph from the original graph determines how many user-item interaction relationships are preserved. With all other hyperparameters held constant, retention rates in the ranges of {0.9, 0.95, 0.99, 0.999} for keeprate are evaluated using the LastFM dataset as a case study. The results are shown in Figure 4:

As depicted in the illustration, the performance of the model's recommendations exhibits a significant variation in response to alterations in the keeprate parameter. When the keeprate is set to 0.99, the model demonstrates its optimal performance. It is essential to recognize that hyperparameter values that stray too far in either direction be it excessively large or small, can lead to a decline in the model's performance. This decline may manifest as underfitting, where the model fails to capture the underlying pattern of the data, or overfitting, where the model becomes overly complex and performs well on training data but poorly on unseen data. Therefore, selecting an appropriate keeprate value is essential for ensuring the model reaches its peak performance potential.

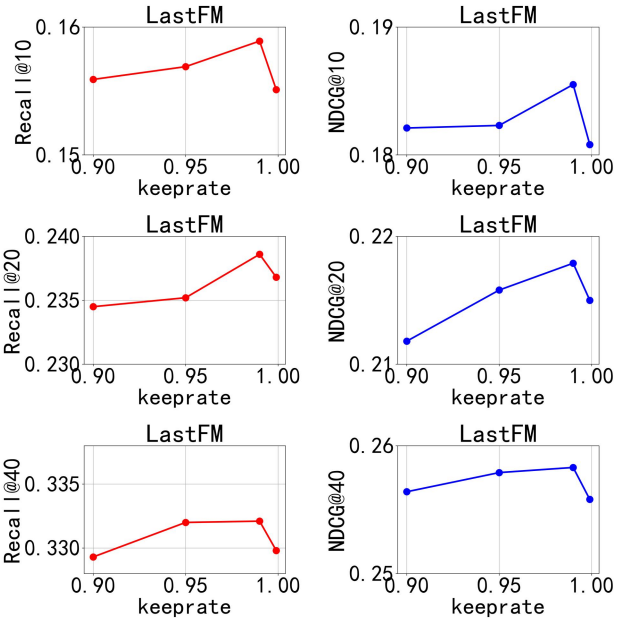

Fig. 3. Performance Comparison of different keeprates

VI. Conclusion

In this paper, we propose a dual-attention fusion self-supervised recommendation framework, DSGRec, based on a graph transformer algorithm. It effectively addresses the issues of sparse data modeling and domain discrepancy stemming from varied user-item interaction patterns in recommendation algorithms. By combining sparse attention and domain-aware attention mechanisms, DSGRec significantly enhances the model's ability to capture the complex relationships between users and items while reducing computational complexity and storage requirements. Furthermore, DSGRec incorporates self-supervised learning, enhancing its robustness and generalization in large-scale, sparse data environments. Experimental results have shown that DSGRec outperformed several existing methods on the LastFM, Yelp, and Ifashion datasets, demonstrating its

effectiveness and practicality. Future work will incorporate real-time data update mechanisms to implement more dynamic recommendations and improve the model's performance in complex real-world scenarios.

REFERENCES

[1] L. Boratto, F. Fabbri, G. Fenu, et al., "Fair Augmentation for Graph Collaborative Filtering," in Proc. 18th ACM Conf. Recommender Syst., pp. 158–168, 2024.

[2] C. Huang, L. Xia, X. Wang, X. He, and D. Yin, "Self-supervised learning for recommendation," in Proc. 31st ACM Int. Conf. Inf. Knowl. Manage. (CIKM), pp. 5136 – 5139, 2022.

[3] Y. Xiang, H. Yu, Y. Gong, et al., "Text Understanding and Generation Using Transformer Models for Intelligent E-commerce Recommendations," ArXiv Preprint 2024, Available: https://arxiv.org/abs/2402.16035.

[4] Y. Dang, E. Yang, G. Guo, et al., "Uniform sequence better: Time interval aware data augmentation for sequential recommendation," in Proc. AAAI Conf. Artif. Intell., vol. 37, no. 4, pp. 4225–4232, 2023.

[5] C. Zhou, et al., "Atrank: An attention-based user behavior modeling framework for recommendation," in Proc. AAAI Conf. Artif. Intell., vol. 32, no. 1, pp.4564-4571, 2018.

[6] Z. Hou, X. Liu, Y. Cen, et al., "GraphMAE: Self-supervised masked graph autoencoders," in Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Min., pp. 594–604, 2022.

[7] W. Huang, Y. Deng, S. Hui, et al., "Sparse self-attention transformer for image inpainting," Pattern Recognition, vol. 145, Art. no. 109897, 2024.

[8] L. Yang, Z. Liu, Y. Dou, et al., "ConsisRec: Enhancing GNN for social recommendation via consistent neighbor aggregation," in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, pp. 2141–2145, 2021.

[9] L. Xia, C. Huang, Y. Xu, et al., "Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation," in Proc. AAAI Conf. Artif. Intell., vol. 35, no. 5, pp. 4486–4493, 2021.

[10] Z. Fan, Z. Liu, Y. Wang, et al., "Sequential recommendation via stochastic self-attention," in Proc. ACM Web Conf., pp. 2036–2047, 2022.

[11] Z. Fan, Z. Liu, J. Zhang, et al., "Continuous-time sequential recommendation with temporal graph collaborative transformer," in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., pp. 433–442, 2021.

[12] Z. Yi and I. Ounis, "A unified graph transformer for overcoming isolations in multimodal recommendation," in Proc. 18th ACM Conf. Recommender Syst., pp. 518–527, 2024.

[13] Z. Yi, X. Wang, and I. Ounis, "A directional diffusion graph transformer for recommendation," ArXiv Preprint 2024, Available: https://arxiv.org/abs/2404.03326.

[14] C. Huang, X. Wang, X. He, et al., "Self-supervised learning for recommender system," in Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, pp. 3440–3443, 2022.

[15] X. Ren, L. Xia, Y. Yang, et al., "SSLRec: A self-supervised learning framework for recommendation," in Proc. 17th ACM Int. Conf. Web Search Data Min., pp. 567–575, 2024.

[16] Z. Hu, G. Xu, X. Zheng, et al., "SSL-SVD: Semi-supervised learning--based sparse trust recommendation," ACM Transactions on Internet Technology (TOIT), vol. 20, no. 1, pp. 1–20, 2020.

[17] W. Wei, C. Huang, L. Xia, et al., "Multimodal self-supervised learning for recommendation," in Proc. ACM Web Conf., pp. 790–800, 2023.

[18] Z. Liu, Y. Chen, J. Li, et al., "Contrastive self-supervised sequential recommendation with robust augmentation," ArXiv Preprint 2021, Available: https://arxiv.org/abs/2108.06479.

[19] T. Liu, C. Xu, Y. Qiao, et al., "News recommendation with attention mechanism," ArXiv Preprint 2024, Available: https://arxiv.org/abs/2402.07422.

[20] Y. Hou, W. Gu, K. Yang, et al., "Deep reinforcement learning recommendation system based on GRU and attention mechanism," Eng. Lett., vol. 31, no. 2, pp. 695-701, 2023.

[21] Y. Zhang, Y. Wang, P. Lan, et al., "Conversational recommender based on additive attention and positional encoding," Journal of Intelligent & Fuzzy Systems, vol. 46, no. 3, pp. 6491–6503, 2024.

[22] C. Wu, F. Wu, S. Ge, et al., "Neural news recommendation with multi-head self-attention," in Proc. 2019 Conf. Empirical Methods Natural Lang. Process. (EMNLP-IJCNLP), pp. 6389–6394, 2019.

[23] A. Milogradskii, O. Lashinin, et al., "Revisiting BPR: A replicability study of a common recommender system baseline," in Proc. 18th ACM Conf. Recommender Syst., pp. 267–277, 2024.

[24] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30–37, 2009.

[25] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in Proc. Int. Conf. Res. Develop. Inf. Retrieval (SIGIR), pp. 165–174, 2019.

[26] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in Proc. 24th Int. Conf. World Wide Web, pp. 111–112, 2015.

[27] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph-based collaborative filtering: A linear residual graph convolutional network approach," in Proc. AAAI Conf. Artif. Intell., vol. 34, pp. 27–34, 2020.

[28] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in Proc. Int. Conf. Res. Develop. Inf. Retrieval (SIGIR), pp. 639–648, 2020.

[29] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in Proc. Web Conf. (WWW), pp. 2320–2329, 2022.

[30] L. Xia, Y. Xu, C. Huang, P. Dai, and L. Bo, "Graph meta network for multi-behavior recommendation," in Proc. Int. Conf. Res. Develop. Inf. Retrieval, pp. 757–766, 2021.

[31] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in Proc. Int. Conf. Res. Develop. Inf. Retrieval (SIGIR), pp. 726–735, 2021.

[32] C. Li, L. Xia, X. Ren, et al., "Graph transformer for recommendation," in Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, pp. 1680–1689, 2023.

[33] R. Ying, R. He, K. Chen, et al., "Graph convolutional neural networks for web-scale recommender systems," in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., pp. 974–98, 2018.

[34] T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang, et al., "Self-supervised learning for large-scale item recommendations," in Proc. Int. Conf. Inf. Knowl. Manag. (CIKM), pp. 4321–4330, 2021.