Received 15 November 2023, accepted 2 January 2024, date of publication 8 January 2024, date of current version 6 March 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3350782*

RESEARCH ARTICLE

# Self-Attention Based Sequential Recommendation With Graph Convolutional Networks

**DEWEN SENG**[1], **JINGCHANG WANG**[1], **AND XUEFENG ZHANG**[1,2]
[1]Key Laboratory of Complex Systems Modeling and Simulation, Hangzhou Dianzi University, Hangzhou 310018, China
[2]School of Information Engineering, College of Science and Technology Ningbo University, Ningbo 315300, China

Corresponding author: Xuefeng Zhang (xuefeng0321@163.com)

This work was supported in part by the Fundamental Public Welfare Research Program of Zhejiang Province under Grant LGF22F020036, in part by the Educational Science Planning Project of Zhejiang Province under Grant GH2021562, in part by the Industry-University Cooperative Education Project of Ministry of Education of China, and in part by the National Experimental Teaching Demonstration Center of Computer Science in China.

**ABSTRACT** Learning embeddings representations of users and items lies at the core of modern recommender systems. Existing methods based on Graph Convolutional Network (GCN) and sequential recommendation typically obtain a user's or an item's embedding by mapping from pre-existing features into better embeddings for users and items, such as ID and attributes. GCN integrates the user-item interaction as the bipartite graph structure into the embedding process, which can better represent sparse data, but cannot capture users' long-term interests. Sequential recommendation seek to capture the "context" of users' activities based on their historical actions, but requires dense data to support it. The goal of our work is to combine the advantages of GCN and sequential recommendation models by proposing a novel Self-Attention based Sequential recommendation with Graph Convolutional Networks (SASGCN). It uses multiple lightweight GCN layers to capture high-order connectivity between users and items, and by introducing ratings as auxiliary information into the user-item interaction matrix, it provides richer information. By incorporating self-attention based methods, the proposed model capture long-term semantics through relatively few actions. Extensive experiments on three benchmark datasets show that our model outperforms various state-of-the-art models consistently.

**INDEX TERMS** Sequential recommendation, transform, graph neural network, user ratings, embedding propagation, collaborative filtering.

## I. INTRODUCTION

Recommender system is an important technology for alleviating information overload on the web, and have been widely deployed in fields such as e-commerce, social networks, music and movie services.

Collaborative filtering (CF) is a common recommendation method that identifies users' interests by analyzing their historical behavior [1]. To improve the accuracy and effectiveness of CF, recent research has explored the integration of GCN. NGCF is one such method, which introduces multiple GCN layers to capture high-order connectivity between users and items [2]. Another recent method, LightGCN further simplifies the model by removing the feature transformation and nonlinear activation parts inherited from GCN in NGCF, achieving model lightweight while still maintaining high accuracy and effectiveness [3].

Sequential recommender systems aim to combine personalized recommendations for users based on their historical activities and the "context" of user behavior, the main challenge is how to capture the high-order dynamics of user behavior succinctly. Recurrent Neural Network (RNN) is a common method for capturing dynamic user information, but RNN sequence models have not been able to attain state-of-the-art results in small-data regimes [4]. Recently, self-attention based Transformer models have achieved state-of-the-art performance and efficiency for Natural Language Processing (NLP) tasks [5], the Transformer model has also

The associate editor coordinating the review of this manuscript and approving it for publication was Xiong Luo.

begun to be applied in sequential recommendation [41], [44]. SASRec replaces the RNN component in traditional sequence recommendation with Transformer, resulting in significantly improved computational efficiency [6].

Inspired by the success of these models, we seek to combine self-attention and GCN for sequential recommendation. we propose a novel Self-Attention based Sequential recommendation with Graph Convolutional Networks, called SASGCN. SASGCN simultaneously achieves the efficiency of parallel computation and the high-order connectivity between users and items. We also noticed that the above models only consider the existence of user-item interactions when representing user and item embeddings, which cannot accurately express the degree of user preference for items. To improve the accuracy of our model, we introduce ratings as auxiliary information when constructing the user-item interaction graph, allowing us to assign weights to different user-item interactions. Importantly, introducing auxiliary information when constructing the interaction graph does not increase model complexity. Specifically, we associate users with items to generate the user-item interaction graph, and introduce ratings as auxiliary information to obtain weights for user-item interactions. We then combine the embeddings learned at different propagation layers with a weighted sum to obtain the graph embedding. Meanwhile, the self-attention part adaptively assigns weights to previous items at each time step. Finally, we map the obtained weights to the graph embedding to obtain the final embedding [3], [6].

To summarize, our work makes the following main contributions:

1) We introduce ratings as auxiliary information into the user-item interaction graph, allowing us to obtain weight information for different users on different items.

2) We propose SASGCN, which combines self-attention and GCN to improve the accuracy and computing efficiency of the model without increasing its complexity.

3) We conduct extensive experimental studies on three real-world datasets. The results demonstrate that our model outperforms other state-of-the-art models in terms of both accuracy and efficiency.

The contents of this article are as follows. The first section is introduction, the second section introduces some related work, the third Section gives the detailed description of SASGCN model, the fourth section presents experiment results and analyses on recommender datasets, and the fifth Section is conclusion and future outlook.

## II. RELATED WORK

Several lines of work are closely related to ours. We discussed the existing work on GCN-based CF methods and sequential recommendation methods.

CF is a classic recommendation algorithm that predicts users' interest in recommended items by analyzing their historical behavior. It is one of the most common method in the field of recommendation systems. Matrix factorization (MF) maps the ID of each user and item to an embedding vector and predicts their interaction through the inner product between them [7]. NeuMF predicts user preferences via Multi-Layer perceptron (MLP) [8].

Despite great success, from CF signals it is difficult to obtain embeddings that satisfy desired properties due to being implicitly captured. In recent years, recommendation algorithms based on graph embeddings have addressed this issue by constructing user-item interaction graphs to explicitly encode CF signals [9], [10]. NGCF is a hybrid model that combines neural networks and graph embeddings, which stacks multiple GCN layers to capture high-order connectivity between user and item nodes, improving recommendation performance. LightGCN is a lightweight improvement over NGCF that removes the feature transformation and nonlinear activation parts inherited from GCN. Since each node in the user-item interaction graph of NGCF only contains ID information, the lightweight operation of LightGCN improves the effectiveness and accuracy of the model [2], [3].

Sequence recommendation refers to predicting the next item of interest for users based on their historical behavior sequence, which is an important research direction in recommendation systems. The main challenge lies in how to handle the long and short-term interest evolution in the user behavior sequence and use it to predict their future behavior. Markov chain (MC) assumes the user's next action can be predicted based on their previous actions [11]. Factorizing Personalized Markov Chain (FPMC) combines MF and item-item transition to capture long-term preferences and short-term transitions respectively [12]. High-order MC can consider the long-term preferences of more users compared to first-order MC [13], [14]. Caser treats the embedding matrix of L previous items as an "image" and applies convolutional operations to extract transitions [15]. Other than MC-based methods, RNN is also commonly used to model user sequences. GRU4Rec uses Gated Recurrent Units (GRU) to model click sequences for session-based recommendation [4].

Transformer is a purely attention-based sequence-to-sequence method that has achieved state-of-the-art performance and efficiency on machine translation tasks which had previously been dominated by RNN-based approaches [5], and is gradually being applied to recommendation systems [17], [18]. SASRec was the pioneering work adopting Transformer for sequence recommendation. It employs self-attention modules to learn the weights of items at different positions in the sequence [6]. STOSA embeds each item as a stochastic Gaussian distribution and introduce Wasserstein distances as self-attention weights to measure the pair-wise relationships between items in the sequence [16], [19]. BERT4Rec employs the deep bidirectional self-attention to model user behavior sequences, predicting the random masked items in the sequence by jointly conditioning on their left and right context [22].

Inspired by LightGCN and SASRec, we seek to build a new Self-Attention based Sequential recommendation model with Graph Convolutional Networks, and introduce ratings as auxiliary information to alleviate the problem of insufficient node information in the user-item interaction graph.

## III. METHOD

In this section, we introduce the proposed SASGCN model, which is illustrated in Figure 1. The SASGCN model consists of four main components: (1) an embedding layer that initializes user and item embeddings, (2) multiple embedding propagation layers that refine the embeddings by incorporating high-order connectivity relations, (3) a self-attention layer with position encoding; and (4) a prediction layer that computes user-item relevance scores.

### A. EMBEDDING LAYER

We give the sequence of item IDs interacted by user $u$, denoted as $S^u = (S_1^u, S_2^u, \ldots, S_{|S^u|}^u)$. To capture the temporal dynamics of user behavior, we consider that recent behaviors better represent the user's current preferences, while distant actions have less impact. Therefore, we define a maximum length $n$ and transform the sequence $(S_1^u, S_2^u, \ldots, S_{|S^u|-1}^u)$ into a training sequence $s = (s_1, s_2, \ldots, s_n)$, where $s_n$ consists of the most recent $n$ actions of the user. If the sequence length is less than $n$, we pad the left side of the sequence with zero vectors until the length is $n$ [6].

We set the user embedding matrix as $\mathbf{E}_u \in \mathbb{R}^{M\times d}$ and the item embedding matrix as $\mathbf{E}_i \in \mathbb{R}^{N\times d}$, that is $\mathbf{e}_u^{(0)} = e_u$ and $\mathbf{e}_i^{(0)} = e_i$, where $d$ is the latent dimension, and $M$ and $N$ denote the number of users and items, respectively. We generate the user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{M\times N}$ from the user-item interaction graph. By introducing the rating of users for items, and each entry $R_{ui}$ is rating if $u$ has interacted with item $i$ otherwise 0.

### B. EMBEDDING PROPAGATION LAYER

We construct the relationship between users and items as a user-item interaction graph, aggregating the features of neighbors as the new representation of a target node through the GCN [20]. Considering that nodes only contain ID and rating information, we aim to minimize unnecessary computations and model burden by adopting the simple weighted sum aggregator instead of the nonlinear activation and feature transformation in traditional GCN. Thus, we adopt a lightweight graph convolution operation [2], [21], defined as follows:

$$\begin{aligned}\mathbf{e}_u^{(k+1)} &= \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}\mathbf{e}_i^{(k)},\\ \mathbf{e}_i^{(k+1)} &= \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}}\mathbf{e}_u^{(k)},\end{aligned} \tag{1}$$

where $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ is the symmetrically normalized term of the graph adjacency matrix, $\mathcal{N}_u$ and $\mathcal{N}_i$ denote the first-hop neighbors of user $u$ and item $i$ [20].

To further enhance the representation capability of our model, we stack $K$ layers of GCN and perform a weighted sum operation on the embeddings obtained at each layer. The weighted sum operation is defined as follows:

$$\mathbf{e}_u = \sum_{k=0}^{K}\alpha_k \mathbf{e}_u^{(k)};\ \mathbf{e}_i = \sum_{k=0}^{K}\alpha_k \mathbf{e}_i^{(k)}, \tag{2}$$

where $\alpha_k \geq 0$ denotes the importance of the $k$-th layer embedding. To avoid unnecessary complexity, we set $\alpha_k$ uniformly as $1/(K+1)$.

### C. SELF-ATTENTION LAYER

The self-attention mechanism has been widely applied in various machine learning fields since its inception. When processing sequences, self-attention can better handle fixed-length inputs than attention mechanisms through position encoding [5]. Given a maximum length $n$, we use a position embedding table $\mathbf{P} \in \mathbb{R}^{n\times d}$, where element $\mathbf{p}_i$ denotes the position embedding for the $i$-th position in a sequence. The final input embedding of sequence $s$ is:

$$\mathbf{E_s} = [\mathbf{e}_1 + \mathbf{p}_1, \mathbf{e}_2 + \mathbf{p}_2, \ldots, \mathbf{e}_n + \mathbf{p}_n], \tag{3}$$

where $\mathbf{e}_n$ is the item embedding $\mathbf{e}_i$.

The self-attention layer mainly consists of multi-head attention, feed-forward network, residual networks and layer normalization [5].

Multi-head attention splits the input embeddings into $h$ different embeddings and performs attention calculations on each of them separately. The multi-head attention on the input embeddings is [22]:

$$\mathbf{H} = \mathbf{concat}\{\text{head}_1, \text{head}_2, \ldots, \text{head}_h\}\mathbf{W}, \tag{4}$$

$$\text{head}_i = \text{Attention}(\mathbf{E_s}\mathbf{W^Q}, \mathbf{E_s}\mathbf{W^K}, \mathbf{E_s}\mathbf{W^V}), \tag{5}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{QK}^T}{\sqrt{d}})\mathbf{V}, \tag{6}$$

where $\mathbf{W}$, $\mathbf{W^Q}$, $\mathbf{W^K}$, $\mathbf{W^V}$ are the weight matrices, *Attention* is the scaled dot-product attention, $\mathbf{Q}$ represents the queries, $\mathbf{K}$ the keys, $\mathbf{V}$ the values, and $\sqrt{d}$ is a regularization term that can prevent gradient vanishing problems and improve the generalization ability of the model. Multi-head attention enables the model to pay attention to information in different subspaces, thus capturing richer feature information.

The feed-forward network consists of two fully connected layers with a ReLU activation in between. This design choice enables the model to capture both linear and nonlinear relationships among items in a sequence. By incorporating nonlinear activation functions, the feed-forward network can better model these nonlinear relationships, thereby improving the model's expressive power and overall performance. The feed-forward network is defined as:

$$\mathbf{F}_i = \mathbf{FFN}(head_i) = \mathbf{ReLU}(head_i\mathbf{W}^{(1)} + b^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}, \tag{7}$$

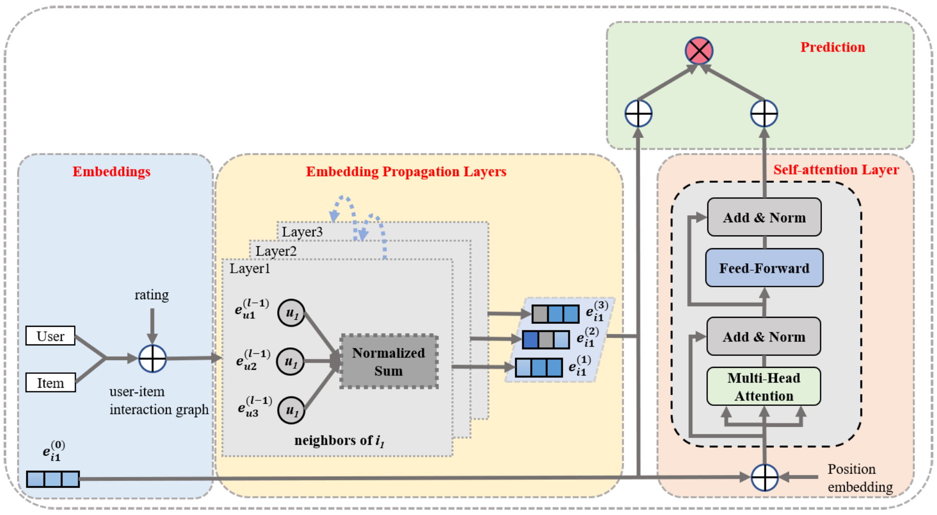

**FIGURE 1.** The framework of the proposed SASGCN model. Obtaining item embedding vectors through the user-item interaction graph with ratings. After multiple layers of GCN propagation and combining with initial vectors, the representations are passed through a self-attention mechanism to capture contextual information before making the final prediction.

where $W^{(1)}$ and $W^{(2)}$ are $d \times d$ matrices and $b^{(1)}$ and $b^{(2)}$ are d-dimensional vectors.

Our model employs residual networks and layer normalization to enable training of deeper models while also improving stability and convergence. The core concept of residual networks is to propagate low-layer features to higher layers using residual connection, which has been shown to improve training of deeper models [23], [24], [42]. Layer normalization is used to normalize inputs across features, which further stabilizes and accelerates neural network training [25]. Batch normalization is not used because the mean and variance of a batch of samples may not adequately represent the mean and variance of the entire population of samples when the batch size is too small [26]. Specifically, assuming the input is a vector $\mathbf{x}$ containing all features of a sample, layer normalization is defined as:

$$\textbf{LayerNorm}(\mathbf{x}) = \alpha \odot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{8}$$

where $\odot$ is an element-wise product, $\mu$ and $\sigma$ are the mean and variance of $\mathbf{x}$, $\alpha$ and $\beta$ are learned scaling factors and bias terms.

## D. PREDICTION LAYER

After obtaining the outputs of the embedding propagation layer and the self-attention layer, for given the first $t$ items, we predict the next item based on $\mathbf{F}_t$. To reduce model size and prevent overfitting, we use a single item embedding $\mathbf{e}_i$. Specifically, we employ an MF layer to predict the relevance of item $i$:

$$r_{i,t} = \mathbf{F}_t \mathbf{e}_i^T, \tag{9}$$

where $r_{i,t}$ is the relevance of item $i$ being the next item given the first $t$ items (i.e., $s1, s2, \ldots, st$ ). By ranking the relevance scores $r_{i,t}$, we can generate top-N recommendations for the user.

## E. MODEL TRAINING

To prevent overfitting during training, we employ dropout by randomly dropping out sequence messages passing into the self-attention layer with probability $p$. Note that dropout is only used during training and must be disabled during testing.

Considering that we truncate or pad the user sequence's last $n$ elements when constructing a fixed-length sequence $s = (s_1, s_2, \ldots, s_n)$, we define $o_t$ as the expected output at time step $t$:

$$o_t = \begin{cases} <pad> & \text{if } s_t \text{ is a padding item} \\ s_{t+1} & 1 \le t < n \\ S^u_{|Su|} & t = n, \end{cases} \tag{10}$$

where $< pad >$ indicates a padding item, and we ignore the terms where $o_t =< pad >$ since we use zero for padding. When the input sequence is $s$, the corresponding sequence $o$ serves as the expected output, and we adopt the binary cross entropy loss:

$$-\sum_{\mathbf{S}^u \in \mathbf{S}} \sum_{t \in [1,2,\ldots,n]} \left[ log(\sigma(r_{o_t,t})) + \sum_{j \notin \mathbf{S}^u} log(1 - \sigma(r_{j,t})) \right]. \tag{11}$$

We employ the Adam optimizer in a mini-batch manner [27], which is a variant of Stochastic Gradient Descent (SGD) with adaptive moment estimation. In each epoch, we randomly generate one negative item $j$ for each time step in each sequence [8], [35].

**TABLE 1. Dataset statistics (after preprocessing).**

| Dataset | Users | Items | Interaction | Sparsity |
|---|---|---|---|---|
| Movielens | 610 | 9724 | 100836 | 0.0169 |
| Amazon-games | 3627 | 27540 | 132515 | 0.0013 |
| Book-crossing | 8001 | 14472 | 138575 | 0.0011 |

### F. COMPLEXITY ANALYSIS

Space Complexity: Our model's learned parameters come from the embeddings and parameters in the multi-head attention, feed-forward network, and layer normalization. The total number of parameters is given by $O(|\mathcal{I} + \mathcal{U}|d + nd + d^2)$, where $\mathcal{I}$ and $\mathcal{U}$ denote the sets of item and user, respectively.

Time Complexity: The computational complexity of our model primarily arises from the embedding matrix, multi-head attention and feed-forward network. The embedding matrix's computation complexity is $O(nd)$, while the multi-head attention and feed-forward network have complexities of $O(n^2d)$ and $O(nd^2)$, respectively. Therefore, the time complexity of our model is $O(nd + n^2d + nd^2)$.

## IV. EXPERIMENTS

We perform experiments on three real-world datasets to assess the performance of our proposed method. Our aim to answer the following research questions:

RQ1: Does SASGCN outperform state-of-the-art models, including sequence recommendation and GCN-based models?

RQ2: How do different hyper-parameter settings, (e.g., dimensions and number of heads) affect SASGCN?

RQ3: What is the impact of various components in the SASGCN architecture?

### A. EXPERIMENTAL SETTINGS

#### 1) DATASETS

To evaluate the effectiveness of SASGCN, we conduct experiments on three benchmark datasets: Amazon-games, Book-crossing, and Movielens, which are publicly accessible and vary in terms of domain, size, and sparsity.

Amazon-games: Amazon-review is a widely used dataset for product recommendation [29], and we select Amazon-games from the collection. This dataset has high sparsity characteristics.

Book-crossing: This dataset was collected from the Book-Crossing community using a web crawler and contains 278,858 users' 1,149,780 ratings on about 271,379 books [30].

Movielens: A widely used benchmark dataset for evaluating collaborative filtering algorithms. We use the version (Movielens-ml-latest-small) that contains approximately 100,836 ratings from 610 users on 9,724 movies [31], [32].

To ensure data quality, we retained users and items with at least ten interactions [33], [39]. We divided each user's historical sequence $S^u$ into three parts: (1) the most recent action $S^u_{|S^u|}$ for testing, (2) the second most recent action $S^u_{|S^u|-1}$ for validation, and (3) the remaining actions for training. Table 1 shows the data statistics. We see that Movielens is the densest dataset, while Amazon-games and Book-crossing are sparse datasets.

#### 2) EVALUATION METRICS

To evaluate the effectiveness of top-$N$ recommendation, we adopt two widely-used evaluation metrics [8], [34], [43]: Recall@$N$ and Normalized Discounted Cumulative Gain (NDCG@$N$). Recall@$N$ measures the fraction of relevant items being retrieved at top-$N$ recommendations out of all relevant items, while NDCG@$N$ evaluates the top-$N$ ranking performance. By default, we set $N = 10$.

#### 3) BASELINES

To demonstrate the effectiveness, we compare our proposed SASGCN with several state-of-the-art recommendation methods:

1) NGCF [2]: A recommendation model based on GCN that learns the embedding vectors of users and items. By stacking multiple GCNs to capture the collaborative signal in high-order connectivities, it finally predict the degree of interest of users to items.

2) LightGCN [3]: A method that removes the feature transformation and nonlinear activation functions in NGCF, reducing unnecessary computations and improving the efficiency and accuracy of recommendation. This also enabling the model to have better generalization capability.

3) Caser [15]: A CNN-based method that captures higher-order Markov Chains by applying convolutional operations on the embedding matrix of the $K$ most recent behavior, and achieves sequential recommendation performance.

4) SASRec [6]: A method that uses self-attention to capture "context" in the sequence instead of traditional sequence recommendation models that use MC and RNN. Due to the self-attention block being suitable for parallel acceleration, SASRec efficiently achieves state-of-the-art recommendation performance.

5) STOSA [16]: A stochastic self-attention sequential model for modeling dynamic uncertainty and capturing collaborative transitivity. By introduce a novel regularization to BPR loss, guaranteeing a large distance between the positive item and negative sampled items.

6) BERT4Rec [22]: A method model user behavior sequences with a bidirectional self-attention network and introduce the Cloze task which predicts the masked items using both left and right context.

#### 4) PARAMETER SETTINGS

For fair comparison, we implement all models using Pytorch with the Adam optimizer [27], the learning rate is set to 0.001, and the batch size is set to 32. The dropout rate is set to 0.5. The maximum sequence length $n$ is set to 200 for the Movielens dataset and 50 for the other two datasets. We consider latent dimension $d$ from

**TABLE 2.** Model performance. The best performing method in each row is boldfaced, and the second best method in each row is underlined.

| | Games | | Books | | Movielens | |
|---|---|---|---|---|---|---|
| | Recall@10 | NDCG@10 | Recall@10 | NDCG@10 | Recall@10 | NDCG@10 |
| NGCF | 0.5105 | 0.3437 | 0.4570 | 0.3002 | 0.6402 | 0.4072 |
| LightGCN | 0.5905 | 0.4123 | 0.5461 | 0.3596 | 0.6719 | 0.4571 |
| Caser | 0.4186 | 0.2555 | 0.4887 | 0.2899 | 0.6501 | 0.4321 |
| SASRec | 0.5980 | 0.4075 | 0.5618 | 0.3624 | 0.6852 | 0.4646 |
| STOSA | <u>0.6219</u> | <u>0.4302</u> | 0.5786 | 0.3877 | 0.7058 | 0.4924 |
| BERT4Rec | 0.6129 | 0.4291 | <u>0.5909</u> | <u>0.3987</u> | <u>0.7175</u> | <u>0.5017</u> |
| SASGCN | **0.6361** | **0.4453** | **0.6210** | **0.4102** | **0.7738** | **0.5178** |

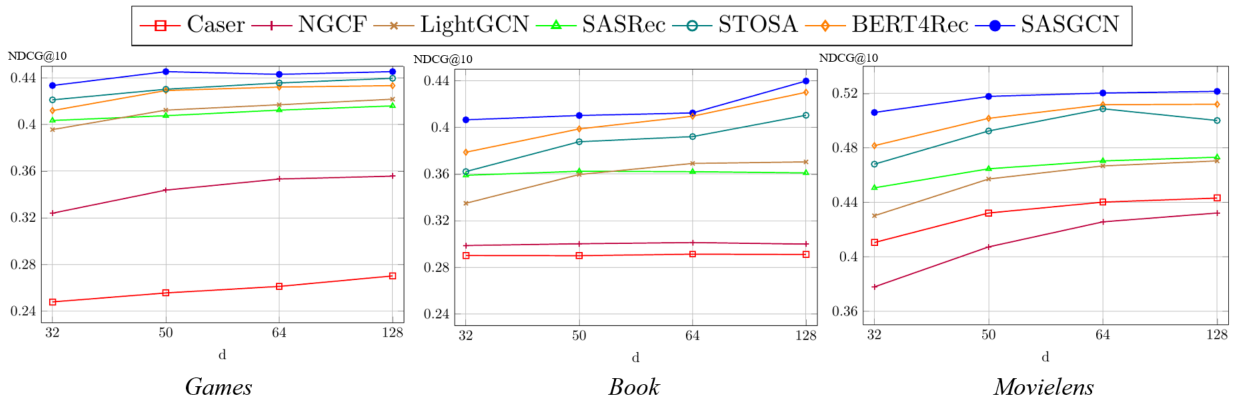

**FIGURE 2.** Effect of the latent dimensionality d on ranking performance (NDCG@10).

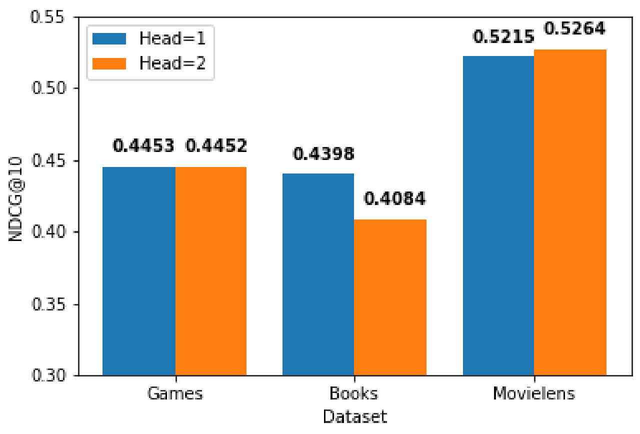

**FIGURE 3.** Effect of the head on ranking performance (NDCG@10).

{32, 50, 64, 128} and analyze the performance of the models under different latent dimensions. All other hyperparameters and initialization strategies are those suggested by the methods' authors. We tune hyper-parameters using the validation set and terminate training if validation performance does not improve for 20 epochs.

### B. PERFORMANCE COMPARISON (RQ1)

Table 2 presents the performance comparison results (with $d = 50$), and we have the following observations:

NGCF and Caser perform relatively poorly across all datasets, while LightGCN improves recommendation performance by introducing lightweight modifications to GCN. SASRec, STOSA, and BERT4Rec perform distinctly better than Caser, suggesting that self-attention mechanism is a more powerful tool for sequential recommendation. STOSA employs stochastic self-attention, and BERT4Rec adopts bi-directional self-attention, both better than SASRec which utilizes the simple self-attention.

SASGCN not only combines the advantages of LightGCN and SASRec, introducing lightweight graph neural networks and self-attention mechanisms into sequential recommendations but also takes into account user rating information on items. According to the result, it is obvious that SASGCN performs best among all methods on three datasets in the terms of all evaluation metrics. It achieve an average improvement of 5.1% in Recall@10 and 3.2% in NDCG@10 compared to the strongest baselines. We conduct t-tests and $p$-value $< 0.02$ indicates that the improvements of SASGCN over the strongest baseline are statistically significant.

While SASGCN demonstrates lower performance on Games and Books datasets than on Movielens dataset. This is due to different levels of sparsity across the datasets, with Books and Games datasets being more sparse, while the Movielens dataset is denser. These results further demonstrate that integrating GCN and auxiliary information into sequence recommendation can effectively mitigate the issue of data sparsity.

### C. PARAMETER IMPACT (RQ2)

In Figure 2, we analyzed the effect of the latent dimension $d$, by showing NDCG@10 of SASGCN with $d$ from

**TABLE 3. Ablation analysis on three datasets.**

| | Games | | Books | | Movielens | |
|---|---|---|---|---|---|---|
| | Recall@10 | NDCG@10 | Recall@10 | NDCG@10 | Recall@10 | NDCG@10 |
| SASGCN | **0.6361** | **0.4453** | **0.6210** | **0.4102** | **0.7738** | **0.5178** |
| w/o rating | 0.6355 | 0.4429 | 0.6144 | 0.3998 | 0.7557 | 0.5023 |
| w/o nonlinear | 0.6352 | 0.4414 | 0.6050 | 0.3954 | 0.7475 | 0.5085 |
| w/o SA | 0.6099 | 0.4179 | 0.5934 | 0.3803 | 0.7228 | 0.4796 |
| w/o GCN | 0.5980 | 0.4075 | 0.5818 | 0.3721 | 0.7262 | 0.4806 |

{32, 50, 64, 128}. We see that our model typically benefits from larger numbers of latent dimensions. For all datasets, our model achieved satisfactory performance with $d = 128$.

In Figure 3, we compared the impact of different "heads" on the model performance with $d = 128$. We see that only the performance on the Movielens dataset was improved when using two heads, while the performance on Games and Books datasets decreased. This might multi-head divides the embedding in $h$ subspaces (each a $d/h$-dimensional space) to capture more diverse information features from different subspaces. Due to the different sparsity levels of the datasets, the information richness contained in the embedding varies. For sparse datasets, only some subspaces contain effective information, while the rest only contain ineffective information, at this time, multi-head cannot capture more diverse feature information but reduces the efficiency and accuracy of information capture.

### D. ABLATION STUDY(RQ3)

To compare the impact of each component in our model on performance, we conducted an ablation study to analyze each component separately. Table 3 presents the performance of our default method and its three variants on all three datasets (with $d = 50$). We introduce these variants and analyze their effects:

(1) SASGCN w/o auxiliary information (ratings): Without user ratings, only consider user-item interactions, similar to Caser and LightGCN, the weights for items interacted with by users are set to 1 and 0 for the rest. This variant performs the best among all variants, indicating that auxiliary information has the smallest impact on the model architecture, but still performs worse than SASGCN.

(2) SASGCN w/o nonlinear: Replace the ReLU activation function in the feed-forward network with the linear activation function Identity. This variant performs worse than (1), indicating that the non-linear activation function in the feed-forward network helps the model better capture the non-linear relationships within the sequences and improve modeling accuracy.

(3) SASGCN w/o SA: Removing self-attention has a significant impact on the dense dataset Movielens. When dealing with richer information, computing different weights for each position in the sequence is beneficial for capturing long-term dependencies in the sequence.

(4) SASGCN w/o GCN: Without GCN, only self-attention is retained. Without graph embedding propagation, higher-order collaborative signals cannot be captured. The results show that GCN can capture richer embedding signals to achieve better recommendation performance, especially on sparse datasets.

## V. CONCLUSION AND FUTURE WORK

In this work, we proposes a Self-Attention based Sequential recommendation with Graph Convolutional Networks, SASGCN. It incorporates ratings as auxiliary information into the user-item interaction graph, and utilizes high-order connectivities in multiple layers of GCN to learn item embeddings. To capture long-term dependencies in the sequence, we introduce position encoding and self-attention mechanism. SASGCN supports parallel computing, thus achieving high efficiency and accuracy. Extensive empirical results on two sparse datasets and one dense dataset show that our model outperforms state-of-the-art baselines. In the future, we plan to introduce richer auxiliary information (e.g. behavior type, social relationship, and item tags) that does not conflict to construct the user-item interaction graph or express the relationship between users and items in the form of a knowledge graph [36], to learn embeddings with richer information. Furthermore, we are interested in exploring the use of bidirection self-attention block for pre-training data to capture the dynamic changes in user interests more accurately [28], and even via reversely pre-training generated data to alleviate data sparsity [37], [38], [40].

## REFERENCES

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, Apr. 2001, pp. 285–295.

[2] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 165–174.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 639–648.

[4] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, *arXiv:1511.06939*.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, p. 111.

[6] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 197–206.

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

[8] X. N. He, L. Z. Liao, H. W. Zhang, L. Nie, and X. Hu, "Neural collaborative filtering," in *Proc. 27st Int. Conf. World Wide Web*, 2017, pp. 173–182.

[9] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. Conf. Uncertainty Artif. Intell.*, 2011, pp. 452–461.

[10] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[11] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *Proc. 11th ACM Conf. Rec. Syst.*, Aug. 2017, pp. 161–169.

[12] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, Apr. 2010, pp. 811–820.

[13] A. Durmus, U. Simsekli, E. Moulines, R. Badeau, and G. Richard, "Stochastic gradient richardson-romberg Markov chain Monte Carlo," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2047–2055.

[14] R. He and J. McAuley, "Fusing similarity models with Markov chains for sparse sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 191–200.

[15] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 565–573.

[16] Z. Fan et al., "Sequential recommendation via stochastic self-attention," in *Proc. 28th ACM Conf.*, 2022, pp. 2036–2047.

[17] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 15908–15919.

[18] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 729–732.

[19] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 83, no. 1, Jan. 2011, Art. no. 016107.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[21] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.

[22] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1441–1450.

[23] B. Liao, J. Zhang, M. Cai, S. Tang, Y. Gao, C. Wu, S. Yang, W. Zhu, Y. Guo, and F. Wu, "Des.-ResNet: A deep spatiotemporal residual network for hotspot traffic speed prediction," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1883–1891.

[24] B. Du, H. Peng, S. Wang, M. Z. A. Bhuiyan, L. Wang, Q. Gong, L. Liu, and J. Li, "Deep irregular convolutional residual LSTM for urban traffic passenger flows prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 972–985, Mar. 2020.

[25] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.

[29] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 43–52.

[30] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proc. 14th Int. Conf. World Wide Web (WWW)*, 2005, pp. 22–32.

[31] GroupLens. (1997). *Movielens Dataset*. [Online]. Available: https://grouplens.org/datasets/movielens/

[32] MovieLens. (1997). *Movielens Website*. [Online]. Available: https://movielens.org/

[33] R. He and J. McAuley, "VBPR: Visual Bayesian personalized ranking from implicit feedback," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 144–150.

[34] J. Ma, C. Zhou, P. Cui, H. Yang, and W. Zhu, "Learning disentangled representations for recommendation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5712–5723.

[35] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, pp. 426–434.

[36] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3549–3568, Aug. 2022, doi: 10.1109/TKDE.2020.3028705.

[37] Z. Liu, Z. Fan, Y. Wang, and P. S. Yu, "Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1608–1612.

[38] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 555–563.

[39] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, Feb. 2019.

[40] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, 2007.

[41] T. Zhang, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, D. Wang, G. Liu, and X. Zhou, "Feature-level deeper self-attention network for sequential recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Beijing, China, Aug. 2019, pp. 4320–4326.

[42] J. Jia and A. R. Benson, "Residual correlation in graph neural network regression," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 588–598.

[43] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *J. Mach. Learn. Res.*, vol. 10, pp. 2935–2962, Dec. 2009.

[44] Y. Chen, Z. Liu, J. Li, J. McAuley, and C. Xiong, "Intent contrastive learning for sequential recommendation," in *Proc. World Wide Web Conf. (WWW)*, 2022, pp. 2172–2182.

**DEWEN SENG** was born in February 1977. He received the Ph.D. degree from the University of Science Technology, Beijing, China, in 2005. He is currently an Associate Professor and the Vice Director of the Software Engineering Institute, Hangzhou Dianzi University. He has (co)authored more than 40 articles. His research interests include artificial intelligence and data mining technology.

**JINGCHANG WANG** was born in March 1999. He is currently pursuing the master's degree with the School of Computer, Hangzhou Dianzi University. His current research interests include recommender systems and attention mechanism.

**XUEFENG ZHANG** was born in March 1980. He received the Ph.D. degree from Kyushu University, Fukuoka, Japan, in 2012. He is currently an Associate Professor with the College of Science and Technology, Ningbo University. He is also a Teacher with Hangzhou Dianzi University. His research interests include artificial intelligence and recommender systems.

• • •