2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI) | 978-1-6654-0898-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICTAI52525.2021.00164

KGAT-SR: Knowledge-Enhanced Graph Attention Network for Session-based Recommendation

Qianqian Zhang, Zhuoming Xu(✉), Hanlin Liu, and Yan Tang
College of Computer and Information
Hohai University
Nanjing, China
E-mail: {qianqian.zhang, zmxu, hanlin.liu, tangyan}@hhu.edu.cn

***Abstract*—As a main task of session-based recommendation, next interaction (item) recommendation aims to recommend the next possible interaction (e.g., click on a song) given a session context (i.e., a list of happened interactions). This task faces the challenge of how to generate accurate recommendations when only the intra-session dependencies are available. Existing recommendation models usually fail to exploit the knowledge about items to model intra-session dependencies. Therefore, this paper addresses the problem of next item recommendation given a session context of an ordered, single-type-action, and anonymous session, and investigates how to effectively model intra-session dependencies by leveraging the knowledge from a knowledge graph (KG). We propose a novel model called Knowledge-enhanced Graph Attention Network for Session-based Recommendation (KGAT-SR). Its core idea is that the knowledge about items from a KG is exploited via knowledge graph attention network to generate a knowledge enhanced session graph (KESG). After the KESG is aggregated via weighted graph attention, the node features and graph topology in the graph are utilized to generate accurate session embedding, which can be used to recommend the next item. Experiments show that KGAT-SR significantly outperforms the state-of-the-art models for next item recommendation. KGAT-SR's source code is available at https://github.com/hu-dske/KGAT-SR.**

Keywords*—session-based recommendation; next item recommendation; knowledge graph; graph attention network

I. INTRODUCTION

Recommender systems (RSs) have proven to be a valuable tool for online users to overcome information/choice overload. Traditional RSs mainly rely on information about historical user-item interactions, but such information is not available in many practical scenarios. Hence session-based recommendation (SR) has recently emerged as a new paradigm of RSs, which typically aims to capture short-term and dynamic user preferences to provide more timely recommendations.

In SR, a session consists of several interactions taken by a user. According to the latest survey [16], typical SR tasks include next interaction recommendation, next partial-session recommendation, and next session recommendation. Next interaction recommendation aims to recommend the next possible interaction (e.g., clicking a song, purchasing a product) in the current session given a session context (i.e., a list of happened interactions), which is usually simplified to predict the next item to interact (termed *next item recommendation*) in the case that there is only one type of interaction. The task of next interaction (item) recommendation faces the challenge of how to generate accurate recommendations when only the intra-session dependencies [16] are available. Such dependencies largely rely on the transitions among interactions within a session, which is thus the key to modeling the dependencies.

Existing SR approaches can be roughly divided into conventional models and deep neural network based models [16]. Conventional SR models, such as [2, 12], predict the possible next interaction by modeling the transitions between consecutive items but ignoring the high-order dependencies within a session. Recently, deep neural networks (DNNs) have been used for SR, and recurrent neural networks (RNNs) and graph neural networks (GNNs) are commonly used DNNs. RNN-based SR models like [4, 5, 8, 10] only model single-way transitions between consecutive items within a session, neglecting complex transitions among distant items. GNN-based SR models such as [11, 19] have shown great expressive power in modeling the complex transitions within a session. However, both RNN- and GNN-based SR models usually fail to exploit the knowledge about items from external knowledge sources, e.g., knowledge graphs (KGs), to effectively model intra-session dependencies (e.g., failing to exploit high-order inter-entity dependencies encoded in the knowledge source), resulting in inaccurate recommendations.

To meet the challenge and overcome the shortcomings of existing SR models, this paper addresses the problem of next item recommendation given a session context of an ordered, single-type-action, and anonymous session, and investigates how to effectively model intra-session dependencies by leveraging the knowledge from a KG. We propose a novel model called Knowledge-enhanced Graph Attention Network for Session-based Recommendation (KGAT-SR). It uses the knowledge graph attention network to learn the high-order inter-entity dependencies encoded in the KG, thereby forming a knowledge enhanced session graph (KESG). KGAT-SR then aggregates the KESG via a weighted graph attention, and utilizes both node features and graph topology via a Readout function to generate accurate session embedding, which can be used to recommend the

978-1-6654-0898-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICTAI52525.2021.00164

next item. Our experiments on two real-world datasets show that KGAT-SR significantly outperforms the state-of-the-art models for next item recommendation.

The contributions of this work are summarized as follows:

- We propose a novel model called Knowledge-enhanced Graph Attention Network for Session-based Recommendation (KGAT-SR). To the best of our knowledge, it is the first session-based recommendation model that exploits external knowledge to enhance session embedding via graph attention networks.
- The Readout function in KGAT-SR can comprehensively exploit node features and graph topology to effectively learn the inherent order of item transition pattern in the session graph.
- Extensive experiments on two real-world datasets show that KGAT-SR significantly outperforms state-of-art models including FGNN, SR-GNN and KSR.

II. Related Work

In this section, we review some related work on next interaction (item) recommendation.

A. Conventional SR Models

Markov chain based SR models are a typical type of conventional SR models. Among others, the factorized personalized Markov chains (FPMC) model [12] combines both a common Markov chain and the normal matrix factorization model, allowing to capture both sequential effects and long term user-taste to provide next-basket recommendation. The personalized ranking metric embedding method (PRME) [2] can jointly consider sequential transitions and user preferences to provide next POI recommendation. As stated in [16], the common shortcomings of the Markov chain based models are that these models ignore the high-order dependencies among interactions and the rigid order assumption is too strong.

B. DNN-based SR Models

RNNs and GNNs are two types of basic DNNs commonly used for session-based recommendation (SR).

1) RNN-based SR models: Hidasi *et al.* [4] applied RNN to SR for the first time and proposed GRU4Rec, an RNN-based model that uses Gated Recurrent Unit to model the high-order dependencies within sessions. The NARM model [8] considers both the user's sequential behavior and main purpose in the current session by incorporating an attention mechanism into RNN, and computes recommendation scores by using a bi-linear matching scheme. The STAMP model [10] replaces the recursive encoder in NARM with an attention layer, which can capture both users' general interests from the long-term memory of a session context and current interests from the short-term memory of the last-clicks. The knowledge enhanced sequential recommender (KSR) [5] integrates the RNN-based networks with key-value memory network semantically enhanced by a knowledge base (KB). But this method fails to exploit high-order inter-entity dependencies encoded in the KB. To sum up, as stated in [19], the common defect of these RNN-based SR models is that they only model single-way transitions between consecutive items within a session and neglect complex transitions among distant items, leading to an inaccurate session embedding.

2) GNN-based SR models: GNNs have recently received more and more attention in the field of SR. The SR-GNN model [19] applies GNN to SR for the first time. In SR-GNN, the session sequence is modeled as a session graph and the GNN is used to capture complex transitions among items. The Full Graph Neural Network (FGNN) model [11] collaboratively considers the sequence order and the latent order in the session graph. However, these GNN-based models fail to exploit external knowledge from a KG (or KB) to model intra-session dependencies. Therefore, we propose a novel model that exploits the external knowledge from a KG to enhance session embedding by modeling intra-session dependencies via graph attention networks.

III. Concepts, Notations and Problem Formulation

In this section we first introduce several concepts and notations, and then formulate the research problem.

Item. The set of all *items* in a recommender system is denoted as $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$. All the embeddings $\mathbf{v}_i \in \mathbb{R}^d$ of the items $v_i \in \mathcal{V}$ are mapped into a unified vector space.

Session. As defined in the survey [16], for session-based recommendation (SR), a *session* is a non-empty bounded list of *interactions* taken by a user, where interactions (e.g., clicking on an item) may be chronologically ordered (in an *ordered session*) or unordered (in an *unordered session*). Sessions can be divided into *single-type-action sessions* that include one type of actions only, and *multi-type-action sessions* that include more than one types of actions (e.g., clicks and purchases). For a specific user's single-type-action sessions, interactions are usually simplified to items. Sessions can be also divided into *non-anonymous sessions* and *anonymous sessions*, depending on whether user information is available. In this paper we study ordered, single-type-action, and anonymous sessions. Formally, a session of length l is denoted as sequence $S = [v_1, v_2, \ldots, v_l]$, which may contain duplicate items, and each item $v_i \in \mathcal{V}$.

Session graph. According to the natural order of the session sequence, the session S can be converted into a weighted directed graph called *session graph* $G_s = (V_s, E_s)$, where $V_s = \{v_1, v_2, \ldots, v_n\}$ represents the node set and each $v_i \in V_s$ is an item in S, whereas E_s represents the edge set and each edge $(v_{i-1}, v_i, w_{(i-1),i}) \in E_s$ indicates that the user interacts with v_{i-1} and v_i consecutively, with $w_{(i-1),i}$ being the weight of the edge, which is defined as the frequency of the occurrence of the edge within G_s. If a node does not

contain self-loops, a self-loop with weight 1 will be added to the node.

Knowledge graph. A *knowledge graph* (KG) is defined as a set of triples $\mathcal{G}=\{(h,r,t)\,|\,h,t\in\mathcal{E},r\in\mathcal{R}\}$, where each triple (h,r,t) indicates that there is a relation $r\in\mathcal{R}$ between the head entity $h\in\mathcal{E}$ and the tail entity $t\in\mathcal{E}$, and each entity in the entity set $\mathcal{E}$ is represented as a node in the graph, each relation in the relation set $\mathcal{R}$ is represented as an edge in the graph. We align the entity $e\in\mathcal{E}$ in the graph $\mathcal{G}$ with the item $v_i\in S$ in the session S to form the *item entity set* in the graph, denoted as $\mathcal{A}=\{e\,|\,e\in\mathcal{E}\text{ can be aligned with }v_i\in S\}$.

Recommendation task. Given a session context of an ordered, single-type-action, and anonymous session S of length l and a knowledge graph $\mathcal{G}$ aligned with the items in the session, the *next interaction (item) recommendation* task is to recommend the next possible interaction (item) v_{l+1} in S by learning intra-session dependencies [16] hidden in the session context and generating a probability distribution vector $\hat{\mathbf{y}}$ over the item set $\mathcal{V}$.

IV. Our Proposed Model

This section describes our KGAT-SR model, including its overall framework, four layers, and model learning.

A. Overview of KGAT-SR

KGAT-SR uses the knowledge about items from a KG to enhance session embedding via graph attention networks. Fig. 1 illustrates the overall framework of KGAT-SR, which consists of the following four layers:

- **KESG Generation**. This layer first aligns the items in the session with the entities in KG to form multi-hop neighborhoods of entities (items), and then employs knowledge graph attention network (KGAT) to learn entity (item) embeddings by capturing the high-order inter-entity dependencies in the neighborhoods. The embeddings and the session graph converted from the session are finally fused through a fusion function to obtain a knowledge enhanced session graph (KESG).
- **Weighted Aggregation on KESG**. This layer employs the L-layer weighted graph attention (WGAT) to perform weighted aggregation on the KESG to generate a post-aggregation KESG.
- **Session Embedding Generation**. This layer utilizes a Readout function to generate a session embedding by comprehensively utilizing the node features and graph topology in the post-aggregation KESG.
- **Probability Prediction**. This layer generates a probability distribution over all items in the item set. The item with the highest probability is the recommended next item.

B. KESG Generation

Once the multi-hop neighborhoods of entities are formed, the KESG Generation layer performs two subsequent processes: multi-hop attentive embedding propagation and fusion.

1) Multi-hop attentive embedding propagation: KGAT-SR uses TransR [9] to obtain embeddings of entities and relations in the KG, i.e., embeddings $\mathbf{e}_h,\mathbf{e}_t\in\mathbb{R}^d$ of head entity h and tail entity t as well as $\mathbf{e}_r\in\mathbb{R}^k$ of relation r of the triple (h,r,t). The propagation process is divided into four steps: information propagation, information aggregation, high-order propagation, and full connection, which are described as follows:

a) Information propagation: This step uses the entities in the KG and their direct neighbors to perform information propagation. Given an entity $h\in\mathcal{A}$, the set of triples whose head entities are h is denoted $\mathcal{N}_h=\{(h,r,t)\,|\,(h,r,t)\in\mathcal{G}\}$. The first-order connectivity structure of h is defined as (1) [17].

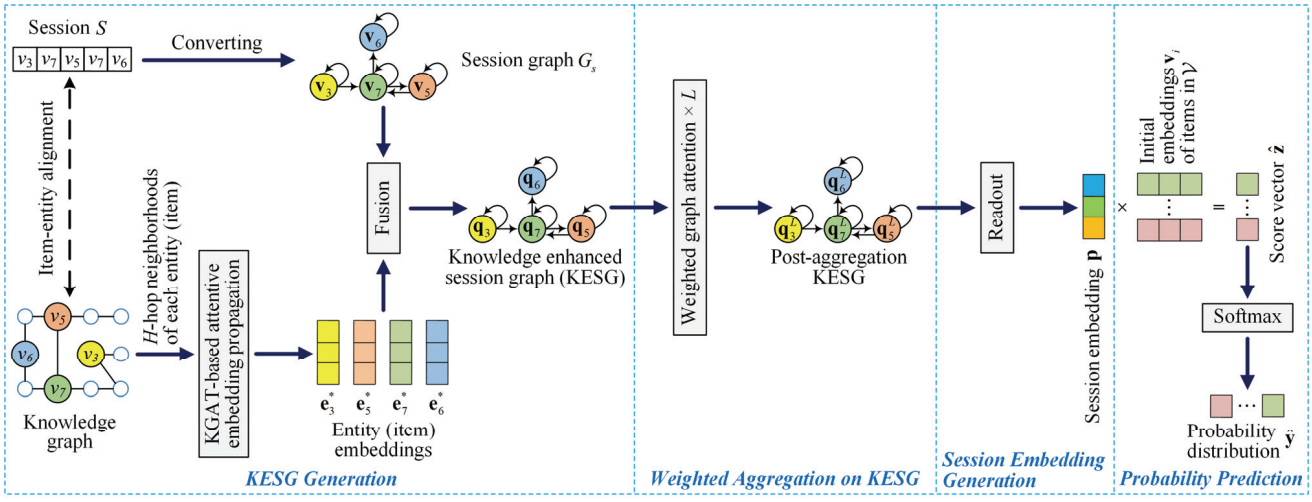


Figure 1. Framework of the proposed KGAT-SR model.

$$\mathbf{e}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t \tag{1}$$

where coefficient $\pi(h,r,t)$ is used to control how much information is propagated from tail entity t to head entity h conditioned to relation r. It is calculated by using the relational attention mechanism [17], which is defined as (2) and (3), where the latter is used to normalize the coefficient.

$$\pi'(h,r,t) = (\mathbf{W}_r\mathbf{e}_t)^\top \tanh(\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r) \tag{2}$$

$$\pi(h,r,t) = \frac{\exp(\pi'(h,r,t))}{\sum_{((h,r',t')\in\mathcal{N}_h)}\exp(\pi'(h,r',t'))} \tag{3}$$

where projection matrix $\mathbf{W}_r \in \mathbb{R}^{k\times d}$ is determined through parameter learning, and $\tanh$ is an activation function.

b) Information aggregation: This step uses the Bi-Interaction Aggregator [17] to aggregate the embedding $\mathbf{e}_h$ of entity h and its first-order connectivity structure $\mathbf{e}_{\mathcal{N}_h}$ to form the next hop representation of the entity, that is $\mathbf{e}_h^{(1)} = f(\mathbf{e}_h, \mathbf{e}_{\mathcal{N}_h})$, where $f(\cdot)$ is defined as (4).

$$\begin{aligned} f(\mathbf{e}_h, \mathbf{e}_{\mathcal{N}_h}) = {} & \text{LeakyReLU}(\mathbf{W}_1(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})) + \\ & \text{LeakyReLU}(\mathbf{W}_2(\mathbf{e}_h \odot \mathbf{e}_{\mathcal{N}_h})) \end{aligned} \tag{4}$$

where LeakyReLU is an activation function, $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^d$ are trainable weight matrices, $\odot$ an element-wise product.

c) High-order propagation: To explore high-order connectivity information, this step stacks more propagation layers to gather information from higher-hop neighbors. At the H-th hop, the representation of entity h is defined as (5).

$$\mathbf{e}_h^{(H)} = f(\mathbf{e}_h^{(H-1)}, \mathbf{e}_{\mathcal{N}_h}^{(H-1)}) \tag{5}$$

where the neighbor node information in the (H–1)-th hop of the entity h is defined as (6).

$$\mathbf{e}_{\mathcal{N}_h}^{(H-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t^{(H-1)} \tag{6}$$

where $\mathbf{e}_t^{(H-1)}$ is the representation of entity t generated from the previous information propagation steps.

d) Full connection: After H-hops propagation, multiple representations $\{\mathbf{e}_h^{(1)}, \mathbf{e}_h^{(2)}, \ldots, \mathbf{e}_h^{(H)}\}$ of entity h are obtained, which emphasize the connectivity information of different orders. They are then concatenated into a vector. This not only enriches the entity embeddings via embedding propagation operations, but also controls the propagation strength by adjusting H. The concatenated vector is defined as (7).

$$\mathbf{e}_h' = \mathbf{e}_h^{(0)} \,\|\, \mathbf{e}_h^{(1)} \,\|\, \ldots \|\, \mathbf{e}_h^{(H)} \tag{7}$$

where $\mathbf{e}_h' \in \mathbb{R}^{d'}$, dimension $d' = d + d^{(1)} + d^{(2)} + \ldots + d^{(H)}$, and $\|$ denotes the concatenation operation. KGAT-SR finally takes linear transformation over $\mathbf{e}_h'$ to obtain the final representation of entity h, which is defined as (8).

$$\mathbf{e}_h^* = \mathbf{W}_a\mathbf{e}_h' + \mathbf{b}_a \tag{8}$$

where weight $\mathbf{W}_a \in \mathbb{R}^{d\times d'}$ and bias $\mathbf{b}_a \in \mathbb{R}^d$ are determined through parameter learning. After the multi-hop attentive embedding propagation, KGAT-SR obtains the embeddings $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \ldots, \mathbf{e}_n^*\}$ of all entities in the set $\mathcal{A}$.

2) Fusion: To effectively learn intra-session dependencies, KGAT-SR uses a fusion function to fuse each entity embedding $\mathbf{e}_i^* \in \{\mathbf{e}_1^*, \mathbf{e}_2^*, \ldots, \mathbf{e}_n^*\}$ with the initial embedding $\mathbf{v}_i$ of the corresponding node in session graph G_s, forming a d-dimensional vector. The fusion function is defined as (9).

$$\mathbf{q}_i = \mathbf{W}_f[\mathbf{e}_i^* \,\|\, \mathbf{v}_i] \tag{9}$$

where weight matrix $\mathbf{W}_f \in \mathbb{R}^{d\times 2d}$ is determined through parameter learning. All nodes in G_s are fused into a d-dimensional vector set $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}$. In this way, the session graph is updated to a KESG.

C. Weighted Aggregation on KESG

KGAT-SR employs the L-layer WGAT to perform weighted aggregation of the nodes in the KESG, which obtains a post-aggregation KESG.

On the basis of vectors $\mathbf{q}_i$, $\mathbf{q}_j$ and weight $w_{i,j}$, KGAT-SR computes an attention coefficient e_{ij} which determines the importance that node j affects node i, as defined in (10).

$$e_{ij} = \text{LeakyReLU}(\mathbf{W}_{att}[\mathbf{W}\mathbf{q}_i \,\|\, \mathbf{W}\mathbf{q}_j \,\|\, w_{i,j}]) \tag{10}$$

where $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_{att} \in \mathbb{R}^{2d+1}$ are weight matrices.

To directly compare the importance of different nodes, a softmax function is applied to converting attention coefficient e_{ij} into a probability form, as defined in (11).

$$a_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}(i)}\exp(e_{ik})} \tag{11}$$

where $\mathcal{N}(i)$ is neighbor nodes of node i. KGAT-SR uses a linear combination to combine the converted attention coefficients with the corresponding neighbors to update the node features, as defined in (12) [11].

$$\mathbf{q}_i' = \mathrm{ReLU}(\sum_{j\in\mathcal{N}(i)} a_{ij}\mathbf{W}\mathbf{q}_j) \tag{12}$$

Multi-head attention can help stabilize the training of the attention layers [14]. Therefore, KGAT-SR applies the multi-head setting, as defined in (13) and (14).

$$\mathbf{q}_i' = \Big\|_{k=1}^{K} \mathrm{ReLU}(\sum_{j\in\mathcal{N}(i)} a_{ij}^k\mathbf{W}^k\mathbf{q}_j) \tag{13}$$

$$\mathbf{q}_i' = \mathrm{ReLU}(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}(i)} a_{ij}^k\mathbf{W}^k\mathbf{q}_j) \tag{14}$$

where K is the number of heads, a_{ij}^k is the weight coefficient calculated by the attention mechanism of head k, and $\mathbf{W}^k$ is the corresponding learning parameter.

Once the forward propagation of the *L*-layer WGAT has been completed, we get the post-aggregation KESG, whose node embeddings are $\{\mathbf{q}_1^L, \mathbf{q}_2^L, \ldots, \mathbf{q}_n^L\}$, $\mathbf{q}_i^L \in \mathbb{R}^d, i = 1,2,\ldots,n$. The post-aggregation KESG will serve as the input to the Session Embedding Generation layer.

D. Session Embedding Generation

Unlike some models that use simple permutation invariant operations, such as Mean, Max, or Sum over all node features, to generate session embeddings, our KGAT-SR model uses SAGPool [7] as a Readout function to generate the session embedding, which can produce a representation of the whole graph (the post-aggregation KESG) based on node features and graph topology. The calculation process of SAGPool is defined as (15)–(17).

$$Z = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Q}\Theta_{att}) \tag{15}$$

$$idx = \text{top-rank}(Z, \lceil kn \rceil),\ Z_{mask} = Z_{idx} \tag{16}$$

$$\mathbf{Q}' = \mathbf{Q}_{idx,:},\ \mathbf{Q}_{out} = \mathbf{Q}' \odot Z_{mask},\ \mathbf{A}_{out} = \mathbf{A}_{idx,idx} \tag{17}$$

where $Z \in \mathbb{R}^n$ is the self-attention coefficient obtained by using graph convolution [6]. $\tilde{\mathbf{A}} \in \mathbb{R}^{n\times n}$ is an adjacency matrix with self-connections, i.e., $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$, and $\tilde{\mathbf{D}} \in \mathbb{R}^{n\times n}$ is the degree matrix of $\tilde{\mathbf{A}}$. $\mathbf{Q} \in \mathbb{R}^{n\times d}$ is the matrix obtained by combining the *d*-dimensional embeddings of the *n* nodes in post-aggregation KESG. $\Theta_{att} \in \mathbb{R}^d$ is the weight parameter. The pooling ratio $k \in (0,1]$ is a hyper-parameter that determines the number of nodes to keep. The top $\lceil kn \rceil$ nodes are selected based on the value of Z. The top-rank function returns the indices of the top $\lceil kn \rceil$ nodes. Subscript $\cdot_{idx}$ is an indexing operation and Z_{mask} is the feature attention mask. $\mathbf{Q}_{idx,:}$ is the row-wise indexed feature matrix and $\mathbf{A}_{idx,idx}$ is the row-wise and col-wise indexed adjacency matrix. $\mathbf{Q}_{out}$ and $\mathbf{A}_{out}$ are the new feature matrix and the corresponding adjacency matrix, respectively. After multiple pooling operations, KGAT-SR obtains session embedding $\mathbf{p} = \mathbf{Q}_{out}$, $\mathbf{p} \in \mathbb{R}^d$.

E. Probability Prediction

KGAT-SR uses the session embedding $\mathbf{p}$ and the initial embeddings of all items in $\mathcal{V}$ to calculate a score vector $\hat{\mathbf{z}}$ for the recommendation, which is defined as (18) [11].

$$\hat{\mathbf{z}} = (\mathbf{W}_{out}\mathbf{p})^\top \mathbf{X}^0 \tag{18}$$

where matrix $\mathbf{X}^0$ is the combination of the initial embeddings of all items in $\mathcal{V}$, and parameter $\mathbf{W}_{out} \in \mathbb{R}^{d\times 2d}$ is determined through parameter learning.

Also, the softmax function is applied to converting $\hat{\mathbf{z}}$ into the probability distribution $\hat{\mathbf{y}}$, as defined in (19).

$$\hat{\mathbf{y}} = \mathrm{softmax}(\hat{\mathbf{z}}) \tag{19}$$

where $\hat{\mathbf{y}}$ indicates the probability that the items in $\mathcal{V}$ will become the next item that the user will interact with.

F. Model Learning

For each session, we define a loss function as cross-entropy of the prediction and its ground truth, as in (20).

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{|\mathcal{V}|}\big(\mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1-\mathbf{y}_i)\log(1-\hat{\mathbf{y}}_i)\big) + \lambda\|\Theta\|_2^2 \tag{20}$$

where $\mathbf{y}_i$ is the one-hot encoding vector of the ground truth item, and Θ is the parameter set for KGAT-SR. *L*2 regularization parameterized by λ on Θ is conducted to prevent overfitting. Finally, we use the Back-Propagation Trough Time (BPTT) algorithm to train the KGAT-SR model.

V. Experimental Evaluation

In this section, we describe our experiments with the purpose of answering the following research questions: i) **RQ1:** Does KGAT-SR outperform state-of-the-art models in terms of recommendation accuracy? ii) **RQ2:** How do different components (i.e., knowledge graph enhanced

component, knowledge graph attention network, Readout function) affect KGAT-SR? iii) **RQ3:** How do different hyper-parameter settings affect KGAT-SR?

A. Datasets

Our experiments used two real-world datasets as follows:

- **MovieLens 1M** [3] contains user ratings for movies (i.e., interaction data) on the MovieLens website.
- **LFM-1b** [20] is a music dataset from the LFM-1b online system, describing users' interaction records on music.

MovieLens 1M and the corresponding KG were released at https://github.com/rexrex9/kb4recMovielensDataProcess by Yu. LFM-1b and the corresponding KG were released at https://github.com/elisabethfischer/KeBERT4Rec by Zhao *et al.* [20]. Similar to [18], we filtered out the items in the datasets that have no corresponding entities in the KGs. We sorted the datasets in ascending order by the timestamp of interaction. The sorted interaction data were divided into multiple sessions at a certain time interval (one day for MovieLens 1M, 30 minutes for LFM-1b). For fair comparison, we filtered out all sessions of length 1 and items that occur less than 5 times in both datasets. Finally, we divided each dataset into a training set and a test set in a ratio of 8:2, and used 10% of the training set as a validation set to tune parameters. The dataset statistics are shown in Table I.

B. Comparison Models

KGAT-SR was compared with seven representative models divided into three categories: conventional models (BPR-MF and FPMC), RNN-based SR models (GRU4Rec, STAMP and KSR), and GNN-based SR models (SR-GNN and FGNN).

- **BPR-MF** [13] is a Bayesian personalized ranking (BPR) optimized matrix factorization (MF) model achieved by applying LearnBPR to MF.
- **FPMC** [12] is the factorized personalized Markov chains model which combines both a common Markov chain and the normal matrix factorization model.
- **GRU4Rec** [4] is an early and well-known sequential recommendation model based on RNN.
- **STAMP** [10] is a hybrid model that constructs two network structures to capture users' general preferences and current interests.
- **KSR** [5] incorporates external knowledge into sequential recommender via key-value memory network.
- **SR-GNN** [19] represents the session sequence as a session graph and uses GNN to model complex transitions among items.
- **FGNN** [11] collaboratively considers the sequential order and the latent order in the session graph, and formulates SR as a graph classification problem.

These comparison models' Python codes released on GitHub were used in our experiments, and the HTTPS hyperlinks of the codes were given in the GitHub library of our KGAT-SR model.

TABLE I. STATISTICS OF DATASETS

Datasets	Statistics			
	#Linked-items	***#Sessions***	***#Entities***	***#Relations***
MovieLens 1M	3,210	31,363	1,125,100	81
LFM-1b	30,658	245,147	214,524	19

C. Parameter Setup

The parameters for the comparison models were set by following their original papers. For KGAT-SR on both datasets, all parameters were initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The embedding dimension d was set to 100. The learning rate was set to 0.0005, and the coefficient λ of $L2$ regularization was set to 10^{-7}. A 3-layer WGAT was applied and each layer has eight heads. The hop number H of attentive embedding propagation was selected in {1, 2, 3, 4} via hyper-parameter analysis experiment (Subsect. F), and was set to 2 for MovieLens 1M and 3 for LFM-1b.

D. Performance Comparison (RQ1)

Similar to the work [5], we uses three popular evaluation metrics [1], *HR*@20, *MRR*@20, and *NDCG*@20, to evaluate the performance of the models. The performance comparison results are shown in Table II, where the best results among all the models are highlighted in boldface, the best results among the seven comparison models are underlined, and the "Improvement" numbers refer to the percentages of performance improvement of KGAT-SR compared to the comparison models with the best performance. We have the following observations from the results:

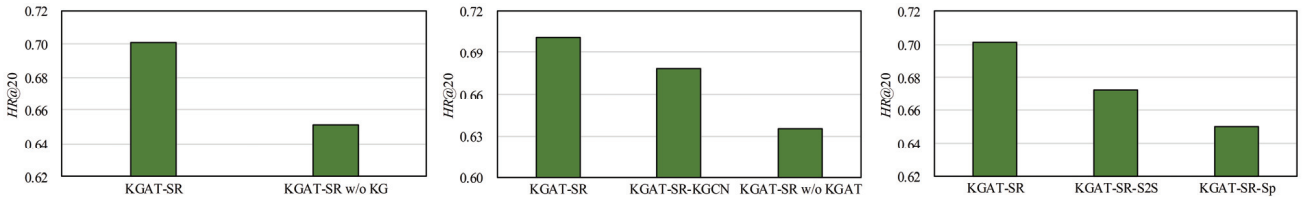
- KGAT-SR achieves the best performance on the two datasets across all the evaluation metrics. This result verifies the advantages of our proposed model.
- The performance of KGAT-SR and KSR is basically better than other models on both datasets. This indicates that the incorporation of external knowledge from KG/KB is conducive to session-based recommendation.
- Regardless of the factor of incorporating external knowledge, the GNN-based models (FGNN and SR-GNN) outperform the RNN-based models (STAMP and GRU4Rec) on both datasets, showing that GNN has advantages over RNN in session-based recommendation.
- The conventional BPR-MF and FPMC models are worse than the DNN-based models on the two datasets, which indicates that DNN is good at session-based recommendation.

E. Influence of Different Components (RQ2)

In order to explore the influences of KGAT-SR's three major components on the model, we have designed several variant models to answer the following three research sub-questions. The results are shown in Fig. 2 and Fig. 3.

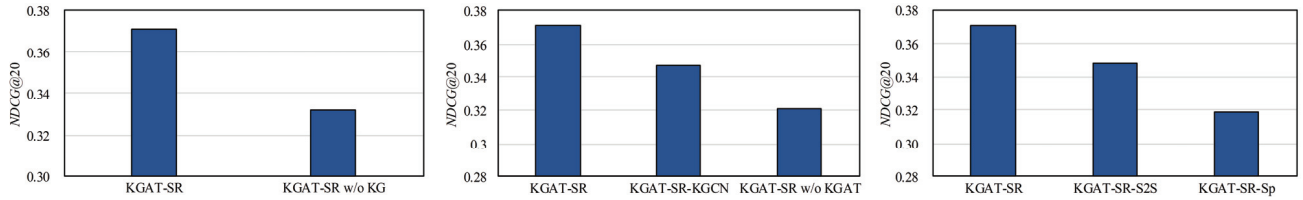
TABLE II. PERFORMANCE COMPARISON OF DIFFERENT MODELS ON TWO DATASETS

Model	MovieLens 1M			LFM-1b		
	HR@20	***MRR@20***	***NDCG@20***	***HR@20***	***MRR@20***	***NDCG@20***
BPR-MF	0.224	0.132	0.156	0.329	0.128	0.154
FPMC	0.482	0.182	0.239	0.582	0.209	0.276
GRU4Rec	0.592	0.217	0.305	0.598	0.280	0.362
STAMP	0.604	0.223	0.309	0.609	0.324	0.395
KSR	0.658	0.268	0.357	0.667	0.438	0.454
SR-GNN	0.627	0.221	0.312	0.633	0.395	0.452
FGNN	0.649	0.291	0.325	0.675	0.420	0.491
KGAT-SR	**0.701**	**0.306**	**0.371**	**0.703**	**0.465**	**0.517**
Improvement (%)	6.53	5.15	3.92	4.15	6.16	5.30



(a) *HR*@20 for RQ2a (b) *HR*@20 for RQ2b (c) *HR*@20 for RQ2c

Figure 2. Performance comparison on MovieLens 1M in terms of *HR*@20.



(a) *NDCG*@20 for RQ2a (b) *NDCG*@20 for RQ2b (c) *NDCG*@20 for RQ2c

Figure 3. Performance comparison on MovieLens 1M in terms of *NDCG*@20.

1) ***RQ2a****: How do the knowledge graph (KG) enhanced component affect KGAT-SR?* The variant model to answer this sub-question is as follows.

- **KGAT-SR w/o KG.** This model is achieved by removing the KG enhanced component including the multi-hop attentive embedding propagation module and the fusion module from KGAT-SR and employing WGAT to perform node representation learning directly on the session graph.

We have the following observations from the results shown in Fig. 2(a) and Fig. 3(a):

- KGAT-SR has a higher accuracy than KGAT-SR w/o KG, which verifies that the incorporation of KG is beneficial to session-based recommendation.

2) ***RQ2b****: How does the knowledge graph attention network (KGAT) affect KGAT-SR?* Two variant models to answer this sub-question are as follows.

- **KGAT-SR w/o KGAT.** This model only uses TransR [9] to obtain embeddings of entities and relations in the KG without using KGAT to perform the multi-hop attentive embedding propagation.
- **KGAT-SR-KGCN.** This model is achieved by replacing the KGAT with the knowledge graph convolutional networks (KGCN) [18] to perform the multi-hop attentive embedding propagation.

We have the following observations from the results shown in Fig. 2(b) and Fig. 3(b):

- Both KGAT-SR and KGAT-SR-KGCN outperform KGAT-SR w/o KGAT, which verifies the importance of the high-order inter-entity dependencies encoded in the KG to modeling the intra-session dependencies.
- KGAT-SR performs better than KGAT-SR-KGCN, demonstrating that KGAT learn the high-order inter-entity dependencies more effectively than KGCN.

3) ***RQ2c****: How does the Readout function affect KGAT-SR?* Below are variant models to answer this sub-question.

- **KGAT-SR-S2S.** This model is achieved by replacing the SAGPool function with the Set2Set [15] function to generate the session embedding.
- **KGAT-SR-Sp.** This model is achieved by replacing the SAGPool function with the Sortpool [21] function to generate the session embedding.

We have the following observations from the results shown in Fig. 2(c) and Fig. 3(c):

- KGAT-SR obtains the best performance among the three models. This result indicates that SAGPool produces better session embedding since it exploits both node features and graph topology, while Set2Set and Sortpool only exploit node features.

F. Hyper-parameter Analysis (RQ3)

In the KESG Generation layer, entity embeddings are generated via the multi-hop attentive embedding propagation. The hop number H of propagation thus affects the generation of entity embeddings. To verify the effect of the hop number, we changed H from 1 to 4 while keeping other hyper-parameters fixed. The results are shown in Table III. We have the following observations from the results:

- Our model achieves the best results at H=2 on MovieLens 1M and at H=3 on LFM-1b, indicating that high-order connectivity information helps improve recommendation performance.
- When H=4, the performance of our model begins to decrease on the two datasets. One possible explanation is that when the hop number is too large, KGAT-SR may capture irrelevant information for entity embeddings.

TABLE III. EFFECT OF THE HOP NUMBERS (H) IN TERMS OF HR@20

Datasets	*H*			
	1	*2*	*3*	*4*
MovieLens 1M	0.672	**0.701**	0.657	0.643
LFM-1b	0.679	0.691	**0.703**	0.675

VI. CONCLUSIONS AND FUTURE WORK

In order to meet the challenge of how to generate accurate next interaction (item) recommendations when only the intra-session dependencies within a session are available and overcome the shortcomings of existing session-based recommendation models, this paper has proposed a novel model called KGAT-SR that exploits knowledge about items from a knowledge graph to enhance session embedding via graph attention networks. The extensive experiments on two real-world datasets have demonstrated that the proposed KGAT-SR model significantly outperforms the state-of-art models for session-based recommendation. This result indicates that the knowledge from knowledge graphs can improve the performance (accuracy) of session-based recommendation. Our future work will focus on investigating how to extend the model to handle multi-type-action sessions and non-anonymous sessions.

REFERENCES

[1] C. C. Aggarwal, "Evaluating Recommender Systems," in Recommender Systems, Springer 2016, pp. 225–254. https://doi.org/10.1007/978-3-319-29659-3_7

[2] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized Ranking Metric Embedding for Next New POI Recommendation," IJCAI 2015, pp. 2069–2075. http://ijcai.org/Abstract/15/293

[3] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," ACM Trans. Interact. Intell. Syst., vol. 5, no. 4, pp. 19:1–19:19, 2016. https://doi.org/10.1145/2827872

[4] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based Recommendations with Recurrent Neural Networks," ICLR (Poster) 2016. http://arxiv.org/abs/1511.06939

[5] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, "Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks," SIGIR 2018, pp. 505–514. https://doi.org/10.1145/3209978.3210017

[6] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," ICLR (Poster) 2017. https://openreview.net/forum?id=SJU4ayYgl

[7] J. Lee, I. Lee, and J. Kang, "Self-Attention Graph Pooling," ICML 2019, pp. 3734–3743. http://proceedings.mlr.press/v97/lee19c.html

[8] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural Attentive Session-based Recommendation," CIKM 2017, pp. 1419–1428. https://doi.org/10.1145/3132847.3132926

[9] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning Entity and Relation Embeddings for Knowledge Graph Completion," AAAI 2015, pp. 2181–2187. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571

[10] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation," KDD 2018, pp. 1831–1839. https://doi.org/10.1145/3219819.3219950

[11] R. Qiu, J. Li, Z. Huang, and H. Yin, "Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks," CIKM 2019, pp. 579–588. https://doi.org/10.1145/3357384.3358010

[12] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," WWW 2010, pp. 811–820. https://doi.org/10.1145/1772690.1772773

[13] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," UAI 2009, pp. 452–461. http://arxiv.org/abs/1205.2618

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," NIPS 2017, pp. 5998–6008. https://arxiv.org/abs/1706.03762v5

[15] O. Vinyals, S. Bengio, and M. Kudlur, "Order Matters: Sequence to sequence for sets," ICLR (Poster) 2016. http://arxiv.org/abs/1511.06391

[16] S. Wang, L. Cao, Y. Wang, Q.Z. Sheng, M.A. Orgun, and D. Lian, "A Survey on Session-based Recommender Systems," ACM Comput. Surv. 54(7): 154:1–154:38 (2021). https://doi.org/10.1145/3465401

[17] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge Graph Attention Network for Recommendation," KDD 2019, pp. 950–958. https://doi.org/10.1145/3292500.3330989

[18] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge Graph Convolutional Networks for Recommender Systems," WWW 2019, pp. 3307–3313. https://doi.org/10.1145/3308558.3313417

[19] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-Based Recommendation with Graph Neural Networks," AAAI 2019, pp. 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[20] W. X. Zhao, G. He, K. Yang, H. Dou, J. Huang, S. Ouyang, and J.-R. Wen, "KB4Rec: A Data Set for Linking Knowledge Bases with Recommender Systems," Data Intell., vol. 1, no. 2, pp. 121–136, 2019. https://doi.org/10.1162/dint_a_00008

[21] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An End-to-End Deep Learning Architecture for Graph Classification," AAAI 2018, pp. 4438–4445. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17146