information

MDPI

Article

Knowledge-Enhanced Graph Neural Networks for Sequential Recommendation

Baocheng Wang and Wentao Cai *

School of Information Science and Technology, North China University of Technology, Beijing 100043, China; wbaocheng@ncut.edu.cn

* Correspondence: 2018309040101@mail.ncut.edu.cn; Tel.: +86-155-7912-3964

Received: 7 July 2020; Accepted: 4 August 2020; Published: 8 August 2020

Abstract: With the rapid increase in the popularity of big data and internet technology, sequential recommendation has become an important method to help people find items they are potentially interested in. Traditional recommendation methods use only recurrent neural networks (RNNs) to process sequential data. Although effective, the results may be unable to capture both the semantic-based preference and the complex transitions between items adequately. In this paper, we model separated session sequences into session graphs and capture complex transitions using graph neural networks (GNNs). We further link items in interaction sequences with existing external knowledge base (KB) entities and integrate the GNN-based recommender with key-value memory networks (KV-MNs) to incorporate KB knowledge. Specifically, we set a key matrix to many relation embeddings that learned from KB, corresponding to many entity attributes, and set up a set of value matrices storing the semantic-based preferences of different users for the corresponding attribute. By using a hybrid of a GNN and KV-MN, each session is represented as the combination of the current interest (i.e., sequential preference) and the global preference (i.e., semantic-based preference) of that session. Extensive experiments on three public real-world datasets show that our method performs better than baseline algorithms consistently.

Keywords: sequential recommendation; knowledge base; graph neural network; memory network

1. Introduction

Recommender systems play an increasingly important role in helping users find items they are interested in by proactively recommending items in a variety of scenarios (e.g., e-commerce, news website and social networking sites). Many existing sequential recommender systems are based on transition probabilities, co-occurrence, item-to-item similarity or recursive neural networks. Despite their effectiveness, user behavior history is complex and the information from browser sessions is limited in many application scenarios; thus, prediction from sequential recommender systems remains a challenging problem [1,2].

The important goal of sequential recommendation is to enhance the quality and efficiency of recommendation. Although many kinds of methods have been developed, recommender systems are still in their early stages of development because of some of the following shortcomings. (1) The Markov chain (MC) [3] is a traditional model which assumes that the user's next behavior is strongly based on previous behavior. However, with an independence assumption, independent combinations of the past components will reduce the performance in the experiment. (2) Recurrent neural networks (RNNs) have improved the performance of the recommender system effectively [4,5]. However, it is very limited in its ability to explicitly capture the complex transitions among distant items or the fine-grained user interest from the sequence of user behavior, but it should be taken into account for a sequential recommender system. (3) Self-attention has been used to model interaction sequence

widely, which is used as a special attention mechanism, and has achieved remarkable results in many scenes (e.g., sentiment analysis and recommender systems) [6]. However, this approach distracts the distribution of attention, reducing local dependencies between adjacent nodes and lacking the ability to learn the contextualized representations of items. (4) Recently, researchers have turned to deep learning models based on knowledge graph for sequential recommendation task. For example, Huang Jin et al. [7] proposed a novel model, namely the knowledge-enhanced sequential recommender (KSR), based on a knowledge graph which consists of two parts, RNNs and key-value memory networks (KV-MNs). By linking the items in interaction sequences with the entities in the external knowledge graph, the RNN component captures a sequential representation for the user, while the KV-MN component captures the representation at the attribute level. However, this approach has limited representation power in explicitly capturing the complex transitions [8,9].

In general, although effective above, we argue that incorporating the external knowledge base into the sequential recommender system can improve the system capability to capture fine-grained preferences and incorporating complex transitions among distant items can improve the accuracy of the item embedding vectors, which are difficult to solve by most of the existing sequential recommendation methods. Traditional approaches always tend to use only limited item information and model only single way transitions between back-to-back items, which makes modeling fine-grained user preferences difficult and ignores the complex transitions among distant items. Therefore, we improve the KSR model by constructing a session graph from interaction sequences and replacing the RNN component with a GNN.

In this work, we propose a knowledge-enhanced recommendation method using graph neural networks and memory networks to tackle the sequential recommendation task and overcome the limitations mentioned above. This consists of a current interest component (i.e., sequential preference) and a global interest component (i.e., attribute-level preference). In the current interest module, we construct directed graphs from a sequence of sessions at first. Based on session graphs, a GNN can obtain the complex transition information between neighbor items and produce the accurate latent vectors of all nodes in the session graph accordingly. However, the priority level of each node in the session graph is different, so we further capture different priorities for each node using the soft-attention mechanism. By doing this, complex transitions between items are considered by the recommender system when making recommendations for users. In the global interest module, we take the external knowledge base into consideration for the sequential recommendation task. The integrated knowledge base should be rich and varied in order to represent different context information in different fields; thus, determining the kind of knowledge to use and the method to represent it is an important step. In this work, items in the interaction sequences are linked to existing knowledge base entities, and the utilization of external knowledge becomes possible for the recommendation task. An external knowledge base consists of lots of triples (i.e., head entity, relation, tail entity), usually corresponding to entities' attribute information. In order to obtain useful representations of external knowledge base information, we use the KB embedding method (i.e., $T_{RANS}E$); thus, entities and relationships are mapped into low-dimensional vectors (i.e., KB embeddings). Due to the structural characteristics of nodes in KBs, we store KB knowledge using KV-MNs which consist of many key vectors and value vectors. Specifically, we set a key vector shared by all users to store a relation embedding that learned from KB knowledge, corresponding to an entity attribute. For specific key vectors, we create different value vectors for different users to store different preferences for the corresponding attribute. As a result, KV-MNs successfully acquire external KB knowledge. We have extensively evaluated our method on three public real-world datasets and compared with many of the baseline approaches using two performance validation metrics. The results in the experiment not only show the improvement of our method over other baseline methods but also verify the effectiveness of our architecture.

To summarize, the main contributions of this work are as follows:

- To improve the representation of session sequences, we propose a novel sequential recommender to fuse the current interest (i.e., sequential preference) and the global preference (i.e., semantic-based preference) effectively.
- To model the current interests of users, we model separated session sequences into session graphs and capture complex transition information between items using graph neural networks.

2. Related Works

2.1. Conventional Recommendation Methods

Traditional recommendation studies are largely focused on collaborative filtering (CF) that is based on the interaction records of users. For instance, matrix factorization (MF) factorizes a user-item rating matrix into two low-rank matrices, each of which contains the latent features of users or items. Yang et al. [10] used the interactions between users; by factorizing social trust network, users are mapped to trustee space and truster space. The Markov chain is a classic method used in recommender systems, which predicts a user's next click according to the last click. By extracting sequential features using a probabilistic decision-tree, Zimdars et al. [11] first use Markov chains to process sequential recommendation tasks. Rendle et al. [3] proposed a hybrid method combining the merit of matrix factorization and Markov chains to implement a next-basket recommendation task.

2.2. Deep-Learning-Based Recommendation Methods

In recent years, researchers have turned to deep learning for sequential recommendation tasks. For example, RNN-based methods have been used to capture the sequential patterns for the sequential recommendation task by Hidasi et al. [4]. Subsequently, neural attentive recommender model (NARM) [5] has been designed to obtain the user's main purpose and sequential pattern accordingly by combining a global and local RNN. More recently, a current attention priority model [12] that employing an attentive net and simple multi-layer perceptron (MLP) networks is proposed in order to completely obtain the current and global interests for user.

The emergence of GNNs leads to unique inspiration for learning the representation for non-Euclidean information. For the sequential recommendation, Wu et al. [1] construct directed graphs from separated session sequences, applying a GNN to generate accurate item embedding vectors. Based on potential vectors of items, the GNN is able to construct more efficient session representations and the next-click item can be inferred.

2.3. Knowledge-Aware Recommendation Methods

With the wide application of internet technology, the availability of a wide variety of knowledge data is increasing [13], such as knowledge information [14] and social interaction information [15]. LibFM is a typical approach that integrates context information into recommender systems. Subsequently, a neural network has been used to improve the modeling of knowledge information, such as RNNs with video semantic embedding [16] and fine-grained attention [17]. Especially, Zhang et al. [18] enhance the performance of deep learning with CF by using knowledge base information.

In this paper, the approach that we proposed is different from previous methods. Our model adopts a graph neural network for capturing complex transitions and incorporates external knowledge base information for enhancing the semantic representation of KV-MN.

3. Research Methodology

In this section, we introduce the proposed recommendation method in detail. We define the task of sequential recommendation at first and show the architecture of the proposed method (as shown in Figure 1).

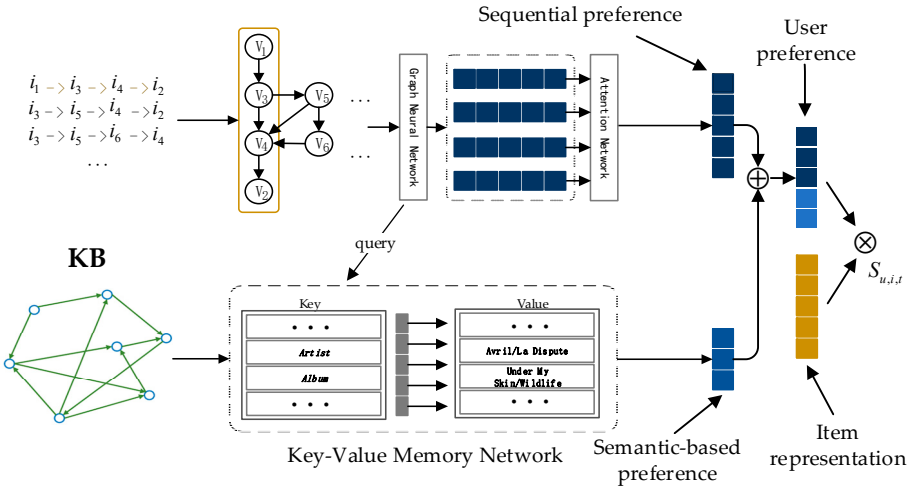

Figure 1. The general architecture of our method. This consists of a key-value memory network (KV-MN) and a graph neural network (GNN). By linking existing external knowledge base entities with items in recommender systems, key-value memory networks are able to incorporate KB knowledge, and the GNN component is used to capture complex transitions. By combining a GNN and KV-MN, the final user preference is a combination of current and global interests.

3.1. Problem Definition

The aim of the sequential recommendation task is to predict which potential items that users will interact with according to past interaction sequences in the near future. Thus, our detailed description for this task is as follows.

In the sequential recommendation task, $V = \{v_1, \ldots, v_t, \ldots, v_m\}$ represents the set containing all unique items in all interaction sequences. A session sequence s containing many clicked events is denoted as $s = [v_{s,1}, \ldots, v_{s,t}, \ldots, v_{s,n}]$, ordered by timestamps, where an interaction event for a user in the session s is defined as $v_{s,i} \in V$. Formally, the aim for the sequential recommendation task is to predict a potential item $v_{s,n+1}$ (i.e., the sequence label) that users will interact with in the near future for the session s. To be exact, for the session s, we generate probabilities $\hat{y}$ for all candidate items accordingly, where an element value of vector $\hat{y}$ represents the prediction score of an item. After this, we set the items with top K values from $\hat{y}$ to be the candidates for recommendation task.

3.2. A GNN-Based Sequential Recommender

Traditional recommendation methods only capture single-way transitions but neglect complex transitions between items. To alleviate this problem, we adopt the graph neural networks as the base sequential recommender to capture rich dependencies from graph-structured data. The GNN is proposed by Scarselli et al. [19], which extends deep learning methods for processing the graph-structured data and is suited for the sequential recommendation task. In this section, we introduce the GNN-based component. We introduce how to construct session graphs from session sequences at first, and then we describe the GNN-based component in detail.

3.2.1. Session Graph Construction

Since interaction sequences are not inherently graph-structured data, we model each session sequence as a directed graph $\varsigma_s = (v_s, \varepsilon_s)$ in which each node corresponds to an item $v_{s,i} \in v_s$ and each edge $(v_{s,i-1}, v_{s,i}) \in \varepsilon_s$ corresponds to two consecutive click events. Since the same item may appear more than once in a sequence in the session sequence, each edge is assigned normalized weight, which is calculated as the occurrence of the edge divided by the outdegree of that edge's start node. For instance, for a session $s = [i_1, i_2, i_3, i_1, i_4]$, the corresponding session graph and connection matrix are

shown in Figure 2. We can apply the two weighted connection matrices with the graph neural network to obtain latent embedding of all items.

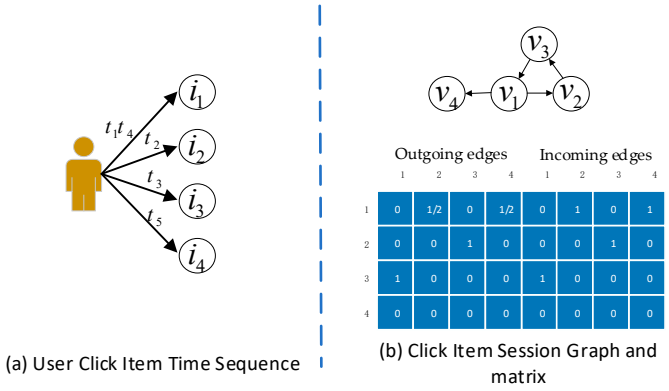

Figure 2. An illustration of a session graph and the connection matrix.

3.2.2. Node Vectors Updating

After this, we introduce the training process of the graph neural network. We embed every item $v \in V$ into a unified low-dimension embedding space, and the vector $\mathbf{v} \in \mathbb{R}^d$ of the node denotes a d-dimensional latent vector corresponding to item v. For each node $v_{s,i}$ in the graph session, the update functions can be formalized as follows:

$$\mathbf{a}_t = \mathbf{M}_{s,i:}\left[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}\right]^\top \mathbf{H} + \mathbf{b} \tag{1}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{a}_t + \mathbf{P}_z \mathbf{v}_i^{t-1}) \tag{2}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{a}_t + \mathbf{P}_r \mathbf{v}_i^{t-1}) \tag{3}$$

$$\widetilde{\mathbf{v}}_t = tanh(\mathbf{W}_o \mathbf{a}_t + \mathbf{P}_o(\mathbf{r}_t \odot \mathbf{v}_i^{t-1})) \tag{4}$$

$$\mathbf{v}_t = (1 - \mathbf{z}_t) \odot \mathbf{v}_i^{t-1} + \mathbf{z}_t \odot \widetilde{\mathbf{v}}_t \tag{5}$$

where $\mathbf{v}_i \in \mathbb{R}^d$ denotes the latent vector of node $\mathbf{v}_{s,i}$, $\mathbf{H} \in \mathbb{R}^{d \times 2d}$ controls the weight, $[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}]$ denotes the list of node vectors in session s, and $\mathbf{b} \in \mathbb{R}^d$ is the bias vector. The connection matrix $\mathbf{M}_s \in \mathbb{R}^{n \times 2n}$ denotes how the information propagation between different nodes and $\mathbf{M}_{s,i:} \in \mathbb{R}^{1 \times 2n}$ are the two rows of blocks in $\mathbf{M}_s$ corresponding to $\mathbf{v}_{s,i}$, and $\mathbf{M}_s$ denotes the concatenation of two adjacency matrices $\mathbf{M}_s^{(\text{out})}$ and $\mathbf{M}_s^{(\text{in})}$, which control weighted connections of outgoing and incoming edges in session graph. $\odot$ denotes the element-wise multiplication operator, and $\sigma(.)$ denotes the logistic sigmoid function, $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_o \in \mathbb{R}^{2d \times d}$, while $\mathbf{P}_z, \mathbf{P}_r, \mathbf{P}_o \in \mathbb{R}^{d \times d}$ are learnable parameters.

When the recommender system starts, the graph neural network simultaneously proceeds to the nodes in the session graph; Equation (1) represents the information interaction between different nodes to obtain the latent vectors based on neighborhoods. Equations (2) and are the update gate and reset gate, which control how information is discarded and retained, respectively. Equation (4) represents the construction of the candidate state, which is based on the reset gate, the current state and the previous state. Equation (5) describes the construction of the final state, which is built on the candidate state, the previous state, and the update gate. Thus, after updating all nodes until convergence, the final node vectors can be obtained.

3.2.3. Generating Sequential Embeddings

After all session graphs are fed into the graph neural network, the latent vectors of all nodes in the session graph are obtained. Each session is represented as an embedding vector $\mathbf{h}_t^u \in \mathbb{R}^d$, which aggregates all node vectors. However, the priority of information in these nodes is different.

Thus, we further represent the sequential preference via the soft-attention mechanism, as described in Equation (6).

$$a_i = \mathbf{q}^\top \sigma(\mathbf{W}^{(1)}\mathbf{v}_n + \mathbf{W}^{(2)}\mathbf{v}_i + \mathrm{c}) \tag{6}$$

$$\mathbf{h}_t^u = \sum_{i=1}^{n} a_i \mathbf{v}_i \tag{7}$$

where $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{W}_1$,$\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ denote the weights of item vectors.

3.3. Augmenting Sequential Recommender with Memory Network

Traditional recommendation models to capture the semantic-based preferences of items are difficult. Meanwhile, capturing the user interests on the semantic level can improve the recommender's performance. Therefore, we propose to incorporate semantic-based information of entities from KB into the sequential recommender and use key-value memory networks to store the KB knowledge, where each memory unit represents a certain type of latent interest for the user; thus, the KV-MN component is integrated with the GNN-based recommender.

3.3.1. Semantic-Based Preference User Interest Modeling

KV-MNs use an external memory for storing data, which splits a memory slot into a key vector and a value vector and associates the key vector with the value vector in a memory slot [20]. Thus, KV-MNs are more powerful in capturing and modeling semantic-based information of items in KBs by storing the attribute information of items in the key vectors and attribute-specific user preferences in value vectors [21].

An item set contains N kinds of unique attribute information. For instance, in the item set of *Movie*, there are unique attributes of director and actor shared by all items in the set. Therefore, we treat the specific user KV-MNs as N pairs of vectors $\left\{\left(\mathbf{k}_1, \mathbf{v}_1^u\right), \ldots, \left(\mathbf{k}_N, \mathbf{v}_N^u\right)\right\}$, where $\mathbf{k}_n \in \mathbb{R}^{L_K}$ represents attribute a and $\mathbf{v}_n \in \mathbb{R}^{\mathrm{Lv}}$ represents the preference of a specific user corresponding to attribute a. For instance, the director attribute of user u is James Cameron. In this way, a public key memory matrix $\mathbf{K} \in \mathbb{R}^{L_K \times A}$ that is shared by all users and a lot of user-specific value memory matrices $\mathbf{V}^u \in \mathbb{R}^{L_v \times A}$ are used for different user accordingly, as shown in Figure 3.

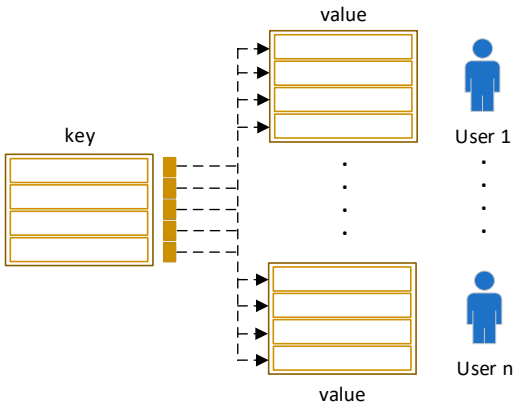

Figure 3. The overview of the KV-MN component. The component consists of two parts, namely a key matrix and a number of value matrices.

By linking external knowledge base entities with items in recommender systems, we can capture semantic information of recommender system items. For an item i, we define e_i as its corresponding entity in KB. Since KB contains a lot of triples originally, we are able to obtain a lot of useful triples. To encode KB information effectively, we plan to obtain a vector $e_i \in \mathbb{R}^{L_E}$ for entity e_i and $r \in \mathbb{R}^{L_E}$ for

relation r by using the effective method $\mathrm{T_{RANS}E}$ [22]. The learned KB embeddings are convenient for subsequent usage.

3.3.2. Write and Read Operations

Write Operation

As long as the recommender system receives a new interaction item i for user u, the system starts the write operation using the entity embedding e_a^i of item i, which updates the user-specific value vectors using the decomposed vectors for e_a^i.

$$\left\{\mathbf{v}_1^u, \ldots, \mathbf{v}_A^u\right\}^{new} \leftarrow \text{WriteOperation}\left(\left\{\left(\mathbf{k}_1, \mathbf{v}_1^u\right), \ldots, \left(\mathbf{k}_A, \mathbf{v}_A^u\right)\right\}^{old}, \mathbf{e}_i\right) \tag{8}$$

$$e_a^i = e_i + r_a \tag{9}$$

where $\mathbf{e}_i$ denotes the embedding representation for item i, $e_a^i \in \mathbb{R}^{L_E}$ denotes the decomposed update vector of item i corresponding attribute a, and r_a represents the relation between the tail entity and head entity. Following [22], we calculate the loss based on the distance $\|e_1 + r - e_2\|$ of a triple $\langle e_1, r, e_2 \rangle$. Thus, the tail entity embedding (i.e., attribute-based embedding) can be approximated by the merging of head entity embedding and relation embedding. For instance, the movie Avatar has the attribute of director corresponding to $e_{Avatar} + r_{directedby} \approx e_{JamesCameron}$.

Following [23], we use a gate vector $z \in \mathbb{R}^A$ to determine the proportion of updated information for each attribute, and the gate weight $z_a \in z$ for each attribute a is shown as follows:

$$z_a = sigmoid\left(v_a^{u\top} \cdot e_a^i\right) \tag{10}$$

With update vector e_a^i and the update weight z_a, each value vector in the value memory matrix V^u is updated.

$$v_a^u \leftarrow (1 - z_a) \cdot v_a^u + z_a \cdot e_a^i \tag{11}$$

After this, the value memory matrix V^u accurately stores the user interests at the attribute level [7].

Read Operation

In order to obtain the preference representation at the attribute level for user u, we take the sequential preference representation $\mathbf{h}_t^u$ from the GNN as the query to the key memory matrix $\mathbf{K}$ at each time t and combine the associated value vector as the return. Since $\mathbf{h}_t^u$ is hard to calculate with the key vectors directly, we implement a nonlinear transformation by utilizing a multiple-layer perceptron, i.e., $\widetilde{\mathbf{h}}_t^u = \text{MLP}(\mathbf{h}_t^u)$. The read operation can be formalized as the following:

$$\mathbf{m}_t^u \leftarrow \text{ReadOperation}\left(\left\{\left(\mathbf{k}_1, \mathbf{v}_1^u\right), \ldots, \left(\mathbf{k}_A, \mathbf{v}_A^u\right)\right\}, \widetilde{\mathbf{h}}_t^u\right) \tag{12}$$

where $\mathbf{m}_t^u$ is a latent vector that is obtained by querying KV-MNs, representing user u's attribute-level preference at time t [7].

3.4. Making Training and Prediction

When the recommender system starts, the system obtains the sequential preference $\mathbf{h}_t^u$ from the graph neural network using Equation (7) at first, and then the transformed hut $\widetilde{\mathbf{h}}_t^u$ is treated as the query to derive the semantic-based preference $\mathbf{m}_t^u$ using Equation (12) correspondingly.

To better infer the user preference, we combine the current representation (i.e., sequential preference) and the global representation (i.e., semantic-based preference) together as $\mathbf{p}_t^u = \mathbf{h}_t^u \oplus \mathbf{m}_t^u$ using a vector concatenation. For the construction of candidate item embedding, we combine the item embedding $\mathbf{v}_n$ in the interaction sequence and the entity embedding $\mathbf{e}_i$ in the external knowledge base

together to obtain the following: $\widetilde{\mathbf{q}}_i = \mathbf{v}_n \oplus \mathbf{e}_i$. After this, we can predict the value of $s_{u,i,t}$ for each candidate item i given the user's preference embedding $\mathbf{p}_t^u$ as follows:

$$s_{u,i,t} = g(u,i,t) = \mathrm{MLP}\left(\mathbf{p}_t^u\right)^\top \cdot \mathrm{MLP}(\widetilde{\mathbf{q}}_i) \tag{13}$$

where MLP(.) denotes a multilayer perceptron. Finally, we can describe the loss function as the cross entropy; this is shown as follows:

$$loss = \sum_{u \in U} \sum_{t=2}^{n_u} \sum_{j \in I_U^-} \log \sigma(g(u,i_t) - g(u,j)) \tag{14}$$

where $\sigma(\cdot)$ denotes the sigmoid function, n_u denotes the length of the interaction sequence of user u in the training set, and I_u^- denotes a small set of sampled negative items that user u has not interacted with.

4. Experiment and Analysis

In this section, we describe the experimental set-up first and then analyze the performance comparison results of different methods. Finally, we further understand our model by conducting a detailed analysis.

4.1. Datasets

We evaluate our method on two types of public datasets (i.e., knowledge base data and recommender systems data). For recommender system data, we adopt three different datasets (i.e., AMAZON book [24], MOVIELENS ml-20m [25], and MOVIELENS ml-1m [25]). For knowledge base data, we use the FREEBASE dump containing lots of triples. Since the MOVIELENS ml-20m dataset is very large, we use only the subset from year 2005 to 2015. Similar to [3,26], we filter out inactive items and users appearing less than k records. The parameter k is set to 10 in MOVIELENS dataset and 3 in AMAZON book datasets. Furthermore, we link items in recommender systems with FREEBASE entities. Specifically, we set item titles as queries and utilize an offline FREEBASE search application programming interface (API) to obtain knowledge base entities. In addition, the interactions associated with the linked items in the filtered datasets are used for recommendations. We classify the original interaction records according to users and sort them by the timestamps ascendingly, and we then set the earliest 70% of the interactions for each user sequence as the training set and set the next 10% of the interactions for each user sequence as the validation set to tune the hyper-parameters, and the remaining 20% of the interactions is set as the test set to report model performance. Since the item set is large, it takes a lot of time to set all the items as candidates. Therefore, similar to [27], each positive item in the test set is paired with 100 sampled items that the user has not associated with (i.e., negative items). To make the sampling real and effective, for the 100 negative items, we pick 50 items randomly and the remaining 50 items according to their popularity.

To train TRANSE, we filter out inactive relations with less than 5000 triples and expand the graph from filtered linked seed entities. We present the detailed statistics in Table 1.

Table 1. Statistics of our datasets.

Dataset	Interactions	Users	Linked Items
Book	828,560	65,125	69,975
ml-20m	5,868,015	61,580	19,530
ml-1m	916,714	6040	3210

4.2. Baseline Algorithms

To verify the validity of our model, we use the following baseline algorithms for comparative analysis:

(1) Factorization-based methods bayesian personalized ranking (BPR-MF) [28] is a classic method optimizing matrix factorization by the Bayesian personalized ranking loss to learn pairwise item rankings.
(2) Factorizing personalized markov chain model (FPMC) [3] is a classic hybrid model that combines Markov chain and matrix factorization to obtain the current and global interests for next-basket recommendation task.
(3) Gated recurrent unit for recommendation (GRU4REC) [4] is a classic model modeling interaction sequences via employing RNNs for the sequential recommendation task.
(4) GRU4REC+ [29] is an enhanced version of GRU4Rec which uses an advanced loss function
(5) NARM [5] captures the pattern of interaction sequences and the main purpose for the user by using RNNs with attention mechanisms.
(6) Short-term attention/memory priority model (STAMP) [12] captures the long-term interests from previous clicks and the current interest from the last clicks for the sequential recommendation task.
(7) Session-based recommendation with graph neural networks (SR-GNN) [1] generates latent item vectors by using a GNN and attention network for the session-based recommendation task.
(8) KSR [7] is a knowledge-enhanced sequential recommender based on the knowledge graph, using gated recurrent unit (GRU) and KV-MN.

4.3. Parameter Setting

In the experiment, the item embedding size is fixed to 50, the KB embedding size L_E with $\mathrm{T_{RANS}E}$ is fixed to 50, and the key vector size L_K and the value vector size L_V are also fixed to 50. For hyper-parameters, we tune them by conducting a grid search on the validation set. Moreover, the learning rate and the batch size are set to 0.001 and 4096, respectively.

4.4. Evaluation Metrics

To compare and verify the performance of our method and the baseline algorithms, we use two classic metrics (i.e., *Recall@K* and *NDCG@K*). Recall@K (R@K) represents the proportion of test cases which has the correctly predicted items in a top K position in a ranking list. NDCG@K (N@K) is the normalized discounted cumulative gain at K, which considers the position of correctly recommended items. By default, we set K to 10.

4.5. Results and Analysis

The results of our method and the baseline algorithms are shown in Table 2.

Table 2. The performance of our method and baseline methods on two metrics over three datasets.

Methods	Ml-20m		Ml-1m		Book	
	R@10	**N@10**	**R@10**	**N@10**	**R@10**	**N@10**
BPR-MF	0.069	0.071	0.082	0.086	0.023	0.013
FPMC	0.071	0.070	0.091	0.098	0.022	0.015
GRU4REC	0.079	0.090	0.089	0.102	0.026	0.015
GRU4REC+	0.081	0.093	0.097	0.112	0.029	0.021
NARM	0.107	0.101	0.116	0.119	0.028	0.028
STAMP	0.106	0.102	0.118	0.110	0.027	0.026
SR-GNN	0.107	0.105	0.121	0.126	0.031	0.029
KSR	0.119	0.121	0.141	0.143	0.039	0.030
OUR MODEL	0.124	0.127	0.150	0.154	0.040	0.029

4.5.1. Observations about Our Model

Firstly, it is obvious that our model achieves almost the best performance in all cases, which verifies the superiority of our model in this field. Secondly, our model performs better than KSR, which illustrates the effectiveness of the construction of the session graph and using graph neural network. Thirdly, our model outperforms SR-GNN, demonstrating the power of adopting semantic-enhanced memory networks in this domain. Fourthly, our model performs better than STAMP, NARM, GRU4REC+, and GRU4REC. One possible reason is that RNN-based models fail to consider sufficient transitions between items in a session. On the contrary, we take complex transitions into account by using a graph neural network. Fifthly, our model outperforms the traditional baselines (e.g., BPR-MF and FPMC) in all cases, which indicates the effectiveness of neural-network-based methods for predicting the next behavior problem.

4.5.2. Other Observations

Firstly, KSR achieves the best performance except in our model, which verifies the effectiveness of introducing an external knowledge base. Secondly, GRU4REC+ outperforms GRU4REC with two evaluation metrics on all datasets, which illustrates the effectiveness of an advanced loss function and sampling strategy. Thirdly, SR-GNN performs better than STAMP, NARM, GRU4REC+, and GRU4REC in nearly all cases. This illustrates the importance of transitions between items in a session. Fourthly, the performance of traditional methods like BPR-MF and FPMC is relatively poor, which verifies the effectiveness of adopting deep learning for a sequential recommendation task.

4.5.3. Model Analysis and Discussion

In this section, we conduct further detailed model analysis in order to better understand the framework of our model.

Impact of Variants of Connection Schemes

Since the process of constructing a session graph from interaction sequences is flexible, we construct a variant connection scheme and compare it with the original connection scheme. Specifically, we first constructed a directed whole item graph from all anonymous session sequences. In the directed whole item graph, each node represents a unique item and each edge represents a directed interaction from one to the other. After this, we can replace the connection matrix with edge weights that are extracted from the directed whole item graph on the basis of the original session graph.

Figure 4 displays the experimental results of applying variants of connection schemes on three datasets. We can observe that both the original scheme and variant scheme outperform SR-GNN, which verifies the effectiveness of KB knowledge, and the original connection scheme outperforms the variant connection scheme. A possible reason is that the variant connection scheme considers the

influence of the whole session graph in addition to the current session graph, which notably affects the integrity of the current session graph.

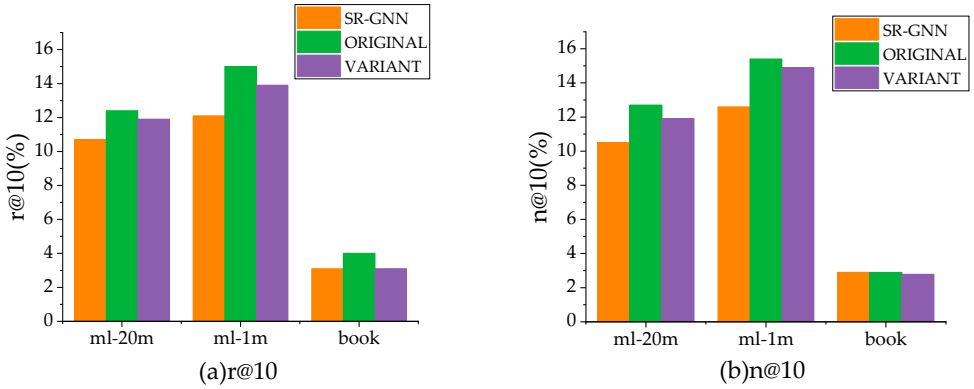

Figure 4. The model's performance under different connection schemes, and figure (**a**) and figure (**b**) are based on recall@k and NDCG@k respectively.

Impact of Varying the Amount of Training Data.

To further discuss the influence of varying amounts of training data on the experiment, we split the complete training data into 20%, 40%, 60%, and 80% for verification and constructed the test sets accordingly. Figure 5 shows the performance of the two methods under tuning of different ratios of dataset (i.e., ml-20m).

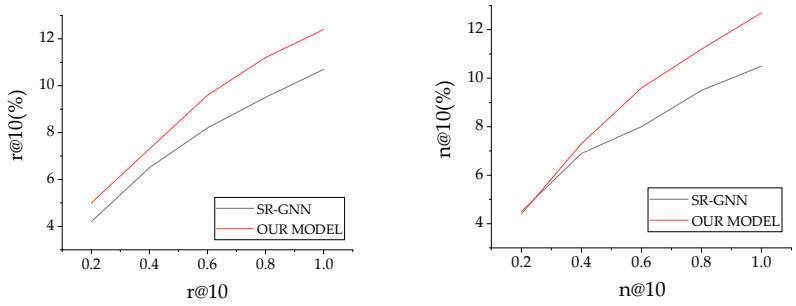

Figure 5. The model's performance of different amounts of training data.

From the figure, it is obvious that our model consistently outperforms SR-GNN with two evaluation metrics, and as the amount of data increases, the values of two evaluation metrics are improved significantly, which indicates that data sparsity negatively impacts performance, and using a hybrid of current and global interests and incorporating KB knowledge can improve the performance of the model.

Impact of Cold Start Scenarios

We further discuss the impact of cold startup on experimental performance. We use the data from the second interaction to the 50th interaction in the ml-20m dataset to observe the performance under cold start scenarios. The experimental results are shown in Figure 6.

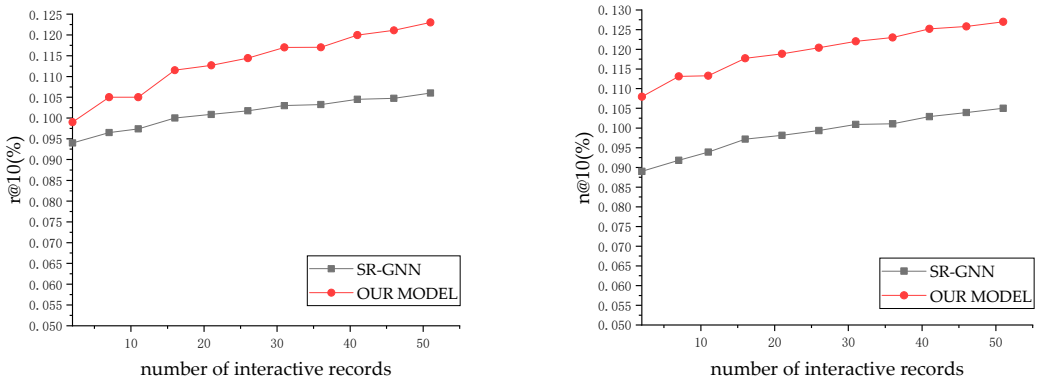

Figure 6. The model's performance under cold start scenarios.

Our method has an inherent advantage for new users regarding the cold start problem; compared with traditional methods (e.g., matrix factorization and collaborative filtering) based on rich user interaction records, our method can use the trained neural network to adapt to new users. As shown in Figure 6, the proposed method on the cold start scenarios also has good performance. With the increase in user interaction records, the performance of our model gradually strengthened, and the experimental results outperformed SR-GNN, which indicates the effectiveness of introducing an external knowledge base and graph neural network.

5. Conclusions and Future Work

In this paper, we devised a novel sequential recommender framework which can effectively capture semantic-based interests and the complex transitions between items. To be exact, we first modeled separated interaction sequences into session graphs to capture complex transitions by using a graph neural network. Next, we linked items in interaction sequences with existing external knowledge base entities and incorporated knowledge base information via KV-MNs. Finally, we combined the current interest (i.e., sequential interest) and the global interest (i.e., semantic-based interest) to better predict users' next actions. Comprehensive experimental comparisons on three public datasets demonstrated that our model can consistently perform better than baseline algorithms in terms of effectiveness. In our future work, we will focus on analyzing, researching, and improving the strategy for constructing session graphs from interaction sequences to better implement the sequential recommendation task.

Author Contributions: Conceptualization, B.W.; data curation, B.W. and W.C.; formal analysis, B.W.; funding acquisition, B.W.; investigation, B.W.; methodology, B.W.; project administration, B.W.; resources, B.W.; software, B.W.; supervision, B.W.; validation, B.W.; visualization, B.W.; writing—original draft, W.C.; writing—review and editing, B.W. and W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The Fudamental Research Funds for Beijing Universities, grant number 110052971921/021.

Acknowledgments: The authors thank the editor and the anonymous reviewers for their valuable suggestions that have significantly improved this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Hilton Hawaiian Village, Honolulu, HI, USA, 27 January–1 February 2019; pp. 346–353.
2. Bai, Y.J.; Jia, S.; Wang, S.; Tan, B. Customer Loyalty Improves the Effectiveness of Recommender Systems Based on Complex Network. *Information* **2020**, *11*, 171. [CrossRef]

3. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
4. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016.
5. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1419–1428.
6. Kang, W.; Mcauley, J. Self-Attentive Sequential Recommendation. In Proceedings of the International Conference on Data Mining, Singapore, 17–19 November 2018; pp. 197–206.
7. Huang, J.; Zhao, W.X.; Dou, H.; Wen, J.-R.; Chang, E.Y. Improving sequential recommendation with knowledge-enhanced memory networks. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 505–514.
8. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A Survey on Knowledge Graph-Based Recommender Systems. *arXiv* **2020**, arXiv:2003.00911.
9. Li, H.; Wang, Y.; Lyu, Z.; Shi, J. Multi-task Learning for Recommendation over Heterogeneous Information Network. *IEEE Trans. Knowl. Data Eng.* **2020**, in press. Available online: https://ieeexplore.ieee.org/abstract/document/9051843/ (accessed on 5 August 2020).
10. Yang, B.; Lei, Y.; Liu, C.M.; Li, W. Social Collaborative Filtering by Trust. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1633–1647. [CrossRef] [PubMed]
11. Zimdars, A.; Chickering, D.M.; Meek, C. Using Temporal Data for Making Recommendations. In Proceedings of the Uncertainty in Artificial Intelligence, Seattle, WA, USA, 2–5 August 2001; pp. 580–588.
12. Liu, Q.; Zeng, Y.; Mokhosi, R.; Zhang, H. STAMP: Short-term attention/memory priority model for session-based recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1831–1839.
13. Verbert, K.; Manouselis, N.; Ochoa, X.; Wolpers, M.; Drachsler, H.; Bosnic, I.; Duval, E. Context-Aware Recommender Systems for Learning: A Survey and Future Challenges. *IEEE Trans. Learn. Technol.* **2012**, *5*, 318–335. [CrossRef]
14. Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; Han, J. Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the Web Search and Data Mining, New York, NY, USA, 24–28 February 2014; pp. 283–292.
15. Zhao, W.X.; Li, S.; He, Y.; Chang, E.Y.; Wen, J.; Li, X. Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 1147–1159. [CrossRef]
16. Gao, J.; Zhang, T.; Xu, C. A Unified Personalized Video Recommendation via Dynamic Recurrent Neural Networks. In Proceedings of the 25th International ACM Conference on Multimedia, Mountain View, CA, USA, 14–19 October 2017; pp. 127–135.
17. Chen, J.; Zhang, H.; He, X.; Nie, L.; Liu, W.; Chua, T. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In Proceedings of the International ACM Sigir Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 335–344.
18. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W.-Y. Collaborative knowledge base embedding for recommender systems. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 353–362.
19. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef] [PubMed]
20. Miller, A.H.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; Weston, J. Key-Value Memory Networks for Directly Reading Documents. In Proceedings of the EMNLP16, Austin, TX, USA, 1–5 November 2016.
21. Liu, F.; Perez, J. Gated End-to-End Memory Networks. In Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; pp. 1–10.

22. Bordes, A.; Usunier, N.; Garciaduran, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 9 December 2013; pp. 2787–2795.
23. Weston, J.; Chopra, S.; Bordes, A. Memory Networks. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 9 May 2015.
24. He, R.; McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the 25th International Conference on World Wide Web, Montréal, QC, Canada, 11–15 April 2016; pp. 507–517.
25. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Sys.* **2015**, *5*, 1–19. [CrossRef]
26. He, R.; Kang, W.-C.; McAuley, J. Translation-based recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 161–169.
27. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.-S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
28. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidtthieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
29. Hidasi, B.; Karatzoglou, A. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In Proceedings of the Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 843–852.