Hindawi
Complexity
Volume 2021, Article ID 6661901, 10 pages
https://doi.org/10.1155/2021/6661901

Research Article

Multi-channel Convolutional Neural Network Feature Extraction for Session Based Recommendation

Zhenyan Ji,[1] **Mengdan Wu**,[1] **Yumin Feng**,[1] **and José Enrique Armendáriz Íñigo**[2]

[1]*School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China*
[2]*Department of Statistics, Computer Science and Mathematics, Public University of Navarre, Pamplona 31006, Spain*

Correspondence should be addressed to Zhenyan Ji; zhyji@bjtu.edu.cn

Received 29 November 2020; Revised 16 February 2021; Accepted 11 March 2021; Published 5 April 2021

Academic Editor: Dan Selisteanu

A session-based recommendation system is designed to predict the user's next click behavior based on an ongoing session. Existing session-based recommendation systems usually model a session into a sequence and extract sequence features through recurrent neural network. Although the performance is greatly improved, these procedures ignore the relationships between items that contain rich information. In order to obtain rich items embeddings, we propose a novel Recommendation Model based on Multi-channel Convolutional Neural Network for session-based recommendation, RMMCNN for brevity. Specifically, we capture items' internal features from three dimensions through multi-channel convolutional neural network firstly. Next, we merge the internal features with external features obtained by a GRU unit. Then, both internal features and external features are merged by an attention mechanism together as the input of the transformation function. Finally, the probability distribution is taken as the output after the softmax function. Experiments on various datasets show that our method's precision and recommendation performance are better than those of other state-of-the-art approaches.

1. Introduction

With the explosive growth of the information in the Internet era, recommendation systems have become an effective solution for users to deal with large amounts of information [1]. In order to have a better user experience, personalized recommendation systems have been applied to many scenarios, including movie recommendation [2, 3], music recommendation [4, 5], online shopping [6, 7], and other settings. In the recommendation scenario, the user behavior is modelled as a session. The session consists of the sequence of clicks performed by the user on items. The first time the user clicks on an item is regarded as the beginning of a session, and the last item of the user's continuous click browsing is the end of the session. Thus, the session contains the time series of user behavior and information between users and items [8, 9].

Traditional recommendation systems are mainly divided into recommendation systems based on collaborative filtering (CF), content-based recommendation systems (CB), and hybrid recommendation systems (HRS) [10]. CF-based recommendation systems build user preference models through the similarity of users or/and items. In addition, the CB recommendation systems state recommendations based on the content of item characteristics [11]. The former does not require contextual features; it only needs to train the matrix factorization model. The latter has good interpretability. In order to combine the advantages of both, HRS emerge to extract information from item attributes [12], users' social networks [13], and item comments [14].

On the other hand, in recent years, deep learning technology has been widely used in recommendation systems [15]. At the same time, powerful cloud computing capabilities have also laid the cornerstone for the development of deep learning [16]. For example, edge computing technology has made it possible to use machine learning technology to achieve intelligent network optimization [17]. Among many neural models, the recurrent neural network [18] approach was the first to be used. Afterwards, the community took into account the rich features of data.

Hence, the user temporal behavior is used in data augmentation [19]. Recently, STAMP [20] and SG-RNN [21] apply graph neural network to capture users' long-term and short-term interests as global interests and the last time the user clicks on the item as the current interest for recommendation.

Although the aforementioned methods achieve greater improvements, they still have some limitations. Firstly, a large number of session recommendation systems are based on users' historical behavior information. Without a large amount of user information, these recommendation systems are not be able to make proper recommendations. Secondly, the sequential features thanks to time stamp are fully captured, but the information between items is ignored.

To overcome the limitations mentioned above, we propose a Recommendation Model based on Mutichannel Convolutional Neural Network (RMMCNN). The main contributions are as follows:

(i) We introduce a multichannel convolutional neural network to extract item information in the context of a session.

(ii) To embed richer features, we use graph neural network to extract sequence features and internal features and then combine them as the final embedding vector representation through an adaptive mechanism.

(iii) Experiments are performed to compare our model with the baseline models. The results indicate that Precision and Mean Average Precision have been increased by at least 0.37% and 0.52%, respectively.

2. Related Work

Conventional recommendation methods include CF, CB, and HRS systems. In recent years, neural networks have greatly improved the performance of recommendation systems, including recurrent neural network, convolutional neural network, and graph convolutional neural network. Recurrent neural network [21–23] can extract users' historical click sequence features. Convolutional neural network [24, 25] can extract different local features of items and generate the corresponding item vector. Graph neural network [21, 26, 27] can learn graph structure data and capture vector embeddings of different nodes. These characteristics enable the neural network to learn more features. Therefore, the neural network can achieve better recommendation performance than conventional recommendation methods.

2.1. Conventional Methods

2.1.1. Collaborative Filtering (CF). Sarwar et al. [28] consider the impact of items on recommendation performance. This work analyzes the user-item matrix to identify different relational items and then uses relational items for indirect calculations. Cheng et al. [29] propose a collaborative filtering method based on user interest sequences. They introduce the similarity of users in the sequence dimension and extract the length of the user's longest common subinterest sequence and the total number of users. The number of common subinterest sequences is used to extract the information hidden in the sequence.

2.1.2. Content Based Recommendation (CB). Putri et al. [30] use the supervised learning method to represent and learn the data of 3700 articles in a vector space and apply a K-Neighbor algorithm for metrics. In order to alleviate the cold start problem, Zhang et al. [31] build the learned feature relationship matrix to extract user preference information hidden in content features. Trinh et al. [32] construct an association matrix between events and the user characteristics content. Concurrently to this, they also combine temporal and spatial relationships together with user interests to make recommendations to friends of key users.

2.1.3. Hybrid Recommendation. Hybrid recommendation algorithms aim to inherit the advantages of CF and CB recommendation algorithms. Rojsattarat and Soonthornphisaj [33] improve the recommendation performance using support vector machines, which map the information to the Euclidean space in order to extract features. Kiewra [34] utilizes the similarity between search and recommendation and uses positive and negative feedback to enhance the range of recommendations. Kolahkaj et al. [35] give importance to certain features of data such as time, location, user's hidden rating, and geographic information location. Then, it combines this information with collaborative filtering, context awareness, and other methods to make recommendations dynamically.

2.2. Deep Learning Methods

2.2.1. Recurrent Neural Network (RNN). Based on the RNN [22], the user's next clicked item can be predicted through similarity. Taking into account the essential characteristics of the item sequence, Long Short-Term Memory (LSTM) is used to capture the similarity between sequences. Xia et al. [23] combine RNN and an attention mechanism to learn about the session, sequence characteristics, and session context information. This fully mines the sequence characteristics of user sessions through recurrent neural network. Wu et al. [21] use the GRU unit to capture the sequence features and combine them with the last item in a session. Then, they are both generated into potential vectors for recommendation.

2.2.2. Convolutional Neural Network (CNN). Cai et al. [24] propose a multi-domain recommendation method based on CNN. It uses the generated user and item preference vectors to predict product ratings through a decomposition machine. Gao et al. [25] establish the CNN to capture the user's sequential features as positive feedback information and obtain negative feedback information through confrontation training, respectively. Afterwards, it combines both

feedbacks to generate action value functions for recommendation.

2.2.3. Graph Neural Network (GNN). GNNs are excellent in node information and graph structure information extraction. Fan et al. [26] propose a GNN framework for user-item graphs and their interactions to model two graphs and heterogeneous intensities. Xian et al. [27] construct the framework that combines the GNN with the repetitive exploration mechanism. It dynamically processes the sequence in a session through the graph structure and captures the complexity between items through the graph neural network. Wu et al. [21] use GNN and the GRU unit to generate a latent vector representation of a session sequence information and apply an attention mechanism to combine global and local user preferences.

3. The Proposed Model

In this section, we firstly present the proposed RMMCNN model. Then, we formulate the problem and introduce the process of our proposed model. Our model includes five steps: (1) knowledge distillation, (2) external feature extraction, (3) internal feature extraction, (4) joint feature extraction, and (5) possibility prediction.

3.1. RMMCNN Framework. Figure 1 introduces the framework of the proposed RMMCNN method. At first, all sessions are fed into a vector space via a directed graph. The items clicked by the user are nodes of the knowledge graph. The direction and connections of nodes in the graph represent the sequence of users that clicked two adjacent items consecutively. If we take into consideration the fact that some users may have clicked the same items in the very same sequence order, we have normalized each edge. We learn the latent vector representation of items through a graph neural network, so that each session will generate the corresponding embedding vector. After the embedding vector generation through the knowledge graph, the embedding vector containing the sequence features is generated by means of the GRU unit. More precisely, the session node embedding vector is propagated among different nodes through the GRU, not only extracting the features of neighboring nodes but also combining these neighboring features as the input of the graph neural network. Following that, the reset gate in the GRU determines whether the information should be kept or dropped, whereas the update gate refreshes all nodes to ensure the convergence of the results. Then, we extract external features and internal features separately for these embedding vectors. Internal features are extracted through a multi-channel convolutional neural network, which has three channels, where each channel has its own weight parameter, and we extract separately the internal features of different dimensions. The results of all channels will be combined through the attention mechanism as the final internal embedding. Finally, the node (resp., item) click probability is generated through linear and softmax transformations.

3.2. Problem Formulation. The goal of a session-based recommendation system is to predict for each session which item is going to be clicked next. Let $V = \{v_1, v_2, v_3, \ldots, v_n\}$ denote the set of all unique items in the sessions. All the sessions are composed of a series of items. Let $s = [v_1^s, v_2^s, v_3^s, \ldots, v_m^s]$ represent a history session sorted by time, where $v_i^s \in V$ represents that the user clicked the item i in session s. In this session, our goal is to predict where the user is going to click next, that is, v_{m+1}^s, in the context of the session.

Next, we are going to describe the steps that we follow to obtain the next clicked item in a session.

3.3. Knowledge Distillation. Recall that, within each session s, each item v is represented as a node in the directed graph $G_s = (V_s, \xi_s)$. The direction of each edge between nodes represents that the user clicked both items consecutively, according to the direction of the edge. Thus, a user in a session s firstly clicks item $v_{m-1}^s \in V$ and, immediately after that, clicks the item $v_m^s \in V$, which we denote as $(v_{m-1}^s, v_m^s) \in \xi_s$. In the directed graph, each item would be embedded into a unified embedding space. Let $N = \{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \ldots, \mathbf{n}_n\}$ denote the item embedding vectors, where $\mathbf{n} \in \mathfrak{R}^{\mathbf{d}}$ indicates the vector embedding of item v and d indicates its dimensionality, respectively. Let $s = [\mathbf{n}_1^s, \mathbf{n}_2^s, \mathbf{n}_3^s, \ldots, \mathbf{n}_m^s]$ denote the session s in the graph G_s^u. For this session s, the proposed RMMCNN will output probabilities $\widehat{y}$ for all the items, where $\widehat{y} = \{\widehat{y}_1, \widehat{y}_2, \widehat{y}_3, \ldots, \widehat{y}_n\}$. The top-K items in $\widehat{y}$ will be chosen as the candidate items.

3.4. External Feature Extraction. The session contains the users' clicked items within a period of time. In order to predict which item the user will click and the user behavior, we need to extract the representations of the session. At the same time, users' interests will change over time, and the temporal characteristics of the session should also be extracted. Therefore, we use the GRU joint attention mechanism to represent these external features.

GRU controls the flow of information through gates. GRU uses two gates, combining the input and forget gate of the LSTM into the update gate. The update gate determines the ratio of the previous value and the current one. The computation formula of the update gate is as follows:

$$\mathbf{z}_t^s = \psi\left(\mathbf{W}_{nz}\mathbf{n}_t^s + \mathbf{W}_{hz}\mathbf{n}_{t-1}^s\right), \tag{1}$$

where $\psi(\cdot)$ denotes the sigmoid function:

$$\psi(x) = \frac{1}{1 + e^{-x}}. \tag{2}$$

The reset gate establishes whether the current candidate state needs to depend on the network previous state and the weight of this dependency:

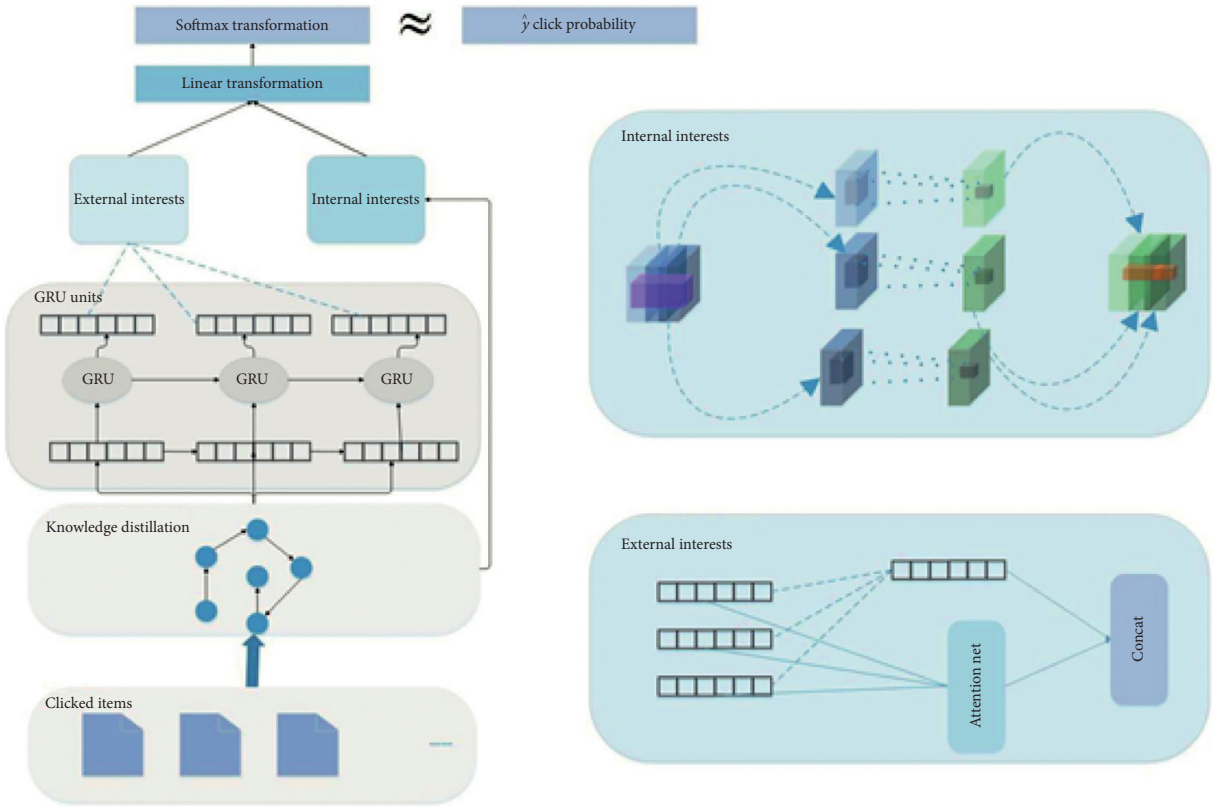

Figure 1: Illustration of the RMMCNN framework.

$$\mathbf{r}_t^s = \psi\left(\mathbf{W}_{nr}\mathbf{n}_t^s + \mathbf{W}_{hr}\mathbf{n}_{t-1}^s\right). \tag{3}$$

We need to estimate another intermediate value, which is the memory value. It is defined as

$$\mathbf{c}_t = \tanh\left(\mathbf{W}_{mn}\mathbf{n}_t^s + \mathbf{W}_{rc}\left(AGG\{(\mathbf{h}_{t-1}\mathbf{r}_t^s)\}\right)\right), \tag{4}$$

which is determined by the memory value in the previous state and the current input value. AGG is the aggregator function. We have chosen to use element-wise multiplication of the vectors. Thus, the state value of the hidden layer is defined as

$$\mathbf{h}_t = AGG\left(\{(1-\mathbf{z}_t^s)\mathbf{c}_t\}\right) + AGG\left(\{\mathbf{z}_t^s\mathbf{h}_{t-1}\}\right), \tag{5}$$

which is the weighted combination of the memory value of the current moment and the previous state value. It is important to note that, in the above formulae, $\mathbf{W}_{**}$ is the corresponding weight matrix.

After the GRU extracts all the features, the session s can be denoted as $\mathbf{v}_t^s = [\mathbf{v}_1^s, \mathbf{v}_2^s, \mathbf{v}_3^s, \ldots, \mathbf{v}_m^s]$.

3.5. Internal Feature Extraction. Following the notations used in Section 3, we use $N = \{\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \ldots, \mathbf{n}_n\}$ to denote the raw input items sequence. We use a multi-channel convolutional neural network to extract the rich internal item features. Besides, we feed $\mathbf{v}_t$ into the neural network for further processing.

We expand the original two-dimension embedding vector into a higher dimension vector by firstly performing a convolution operation, which considers the difference of the internal features extraction with different convolution kernels. Our approach is loosely based on the RGB image processing, and we capture the different dimension features with a multi-channel convolutional neural network that we use to build the future maps. We set the convolution kernel to [1, 1, 1, 1], [1, 1, 1, 2], and [1, 3, 1, 1], respectively. Thus, the three convolution channel results can be expressed as $\widetilde{v}_R^t$, $\widetilde{v}_G^t$, and $\widetilde{v}_B^t$.

Then we merge these future maps with a linear transformation.

$$s_c = f\left(\widetilde{v}_R^t, \widetilde{v}_G^t, \widetilde{v}_B^t\right), \tag{6}$$

where $f(\cdot)$ is a linear transformation function. Once we extract the features, we perform a dimension reduction for subsequent processing to obtain the vector $\widetilde{s}_c$.

On the other hand, we extract the last item as v_l^s, which is denoted as v_m^s; that is, $v_l^s = v_m^s$.

3.6. Joint Feature Extraction. In order to maximize the representation of information, we aggregate the external and internal features together. Firstly, we aggregate all node embedding vectors and the last item:

$$\widetilde{s}_e = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{7}$$

where $\alpha_i = \mathbf{v}^T\sigma(\omega_1\mathbf{v}_l^s + \omega_2\mathbf{v}_i^s + b)$, $i \in [1, m]$, and $\mathbf{v} \in \Re^{d_1}$ controls the weights of item vectors.

Then, we aggregate the result into the final feature. After this, we compute the hybrid embedding vector $\widetilde{s}_f$ through function transformation over the combination of the last clicked item and the external features:

$$\widetilde{s}_f = \boldsymbol{\beta}[\widetilde{s}_e; \widetilde{s}_c], \tag{8}$$

where matrix β compresses two combined embedding vectors into the latent space $\mathfrak{R}^d$.

3.7. Probability Prediction. After obtaining the representation of the session, we calculate the score $\widehat{g}_i^s$ as follows:

$$\widehat{g}_i^s = \widetilde{s}_f^T \boldsymbol{\omega}_3 v_i^s, \tag{9}$$

where ω_3 is the corresponding conversion dimension matrix and $\widehat{g}_i^s \in \mathfrak{R}^n$ represents the similarity scores with regard to each candidate item.

The score is transformed through the softmax function in the following:

$$\widehat{y}_i = \text{softmax}(\widehat{g}_i^s), \tag{10}$$

where $\widehat{y} \in \mathfrak{R}^n$ denotes the next clicked item probability for a given user in the context of the given session.

In our work, we adopt the cross-entropy as the loss function, and it is defined as follows:

$$\text{Loss}(\widehat{y}) = -\sum_{i=1}^{n} y_i \log(\widehat{\mathbf{y}}_i) + (1 - \mathbf{y}_i)\log(1 - \widehat{\mathbf{y}}_i), \tag{11}$$

where $\mathbf{y}$ denotes the user actual clicked items in the session.

The recommendation model can be trained through the backpropagation algorithm, and then the parameters in the model can be updated. In this process, we use the Adam optimizer [36] to train the parameters in the RMMCNN model.

4. Experiments and Analysis

In this part, we conduct three groups of experiments on two real world datasets. The datasets come from the RecSys 2015 Challenge called Yoochoose and CIKM Cup 2016 Challenge called Diginetica. The first experimental setup compares the recommendation performance of different models, whereas second group experiment compares the recommendation performance of different session embedding methods. Finally, the third group compares the recommendation performance of different evaluation criteria. Our experiments are based on TensorFlow 1.4.0 and Python 3.6.

4.1. Experiment Settings

4.1.1. Datasets. Yoochoose contains a series of click events of users on e-commerce websites, and these click events can be used to predict whether the user intends to click a certain product. Diginetica contains a large amount of information such as searches, logs, product data, and transaction data. In this paper, we only use transaction data.

Considering the existence of some noisy data in the sessions, we will filter out those values whose session length is 1 and those items that have been clicked fewer than 5 times, following [20, 21]. Then, we separate the two datasets and divide them into a training dataset and a test dataset, respectively. The reader is referred to Table 1 for a more detailed description. Considering that the Yoochoose dataset is quite large, we sort the sequences in the Yoochoose dataset and obtain the latest fractions 1/64 and 1/4 of the entire sequence according to time. We refer to them as Yoochoose 1/4 and Yoochoose 1/64.

4.1.2. Data Availability Statement. There are two datasets used in this paper: one is Yoochoose and the other is Diginetica. The Yoochoose dataset comes from the 2015 ACM RecSys Challenge. The content is a series of click events performed by users during a typical session in an e-commerce website. The data files include training data files and test data files. The former contains click events and purchase events, and the latter contains files. Each click event contains the session ID, timestamp, item ID, and item category. The Yoochoose dataset can be accessed at https://2015.recsyschallenge.com/challenge.html. The Diginetica dataset comes from CIKM Cup 2016. The dataset contains product data and transaction data. In this paper, we only use transaction data. The Diginetica dataset can be obtained from the following link: https://competitions.codalab.org/competitions/11161#learn_the_details-data2.

4.2. Baselines. In order to measure the performance of the proposed model, we compare the model with the following baseline algorithms.

(i) **NARM** [37] uses an attention mechanism to obtain the features in the hidden state to enhance the original information, which emphasizes the main purpose of the user in the current session. It proposes a neural attention recommender to solve the problem of lack of user purpose analysis in a session-based recommendation setting. NARM proposed a hybrid encoder to simulate the user's sequential behavior, capture the user's main purpose in the conversation, and merge this information as the final user behavior information representation.

(ii) **STAMP** [20] combines the current session information with the last clicked item in the current session. This mainly solves the problem of user behavior prediction based on anonymous sessions. It considers the impact of the user's current operation on the next clicked item as a tradeoff with the short-term memory model. It combines the short-term attention model with the original long-term memory model to extract the current and long-term user interest and generates the final interest of the user.

(iii) **SRGNN** [21] uses graph neural networks and GRU units to generate node latent vector representations.

Table 1: The statistics of the datasets.

Datasets	Yoochoose 1/4	Yoochoose 1/64	Diginetica
No. of clicks	7,980,529	565,552	982,961
No. of items	30,660	17,694	43,097
No. of train sessions	5,917,746	369,859	719,470
No. of test sessions	55,898	55,898	60,858
Average length (no. of clicks/session)	5.71	6.16	5.12

This approach tries to solve the problem of accurate user vector generation. SRGNN models user behavior as a graph structure data and captures item conversion information with a graph neural network. Then, the final embedding vector is generated through a linear transformation and the recommendations are made based on user's clicked items sequence and the last clicked item in the session.

4.3. Evaluation Metrics. We use often-used Precision (P) and Mean Reciprocal Rank (MRR) evaluation metrics to evaluate the performance of the RMMCNN model.

P@20: P@K measures predictive accuracy in recommendation systems. P@K describes the ranking ratio of recommended items accuracy in the recommendation lists, and it is defined as follows:

$$P@K = \frac{c_{\text{hit}}}{|C|}, \tag{12}$$

where $|C|$ denotes the total count of test data and c_{hit} represents the count of the hit data in the top-K ranking list.

MRR@20: MRR@K measures the accuracy of recommended clicked items, ordered by the probability of correctness. Given $K = 20$, if the right clicked item is suggested in apposition greater than 20, it will be set to zero:

$$MRR@K = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{1}{\text{rank}_i}, \tag{13}$$

where rank_i denotes users' first item ranking position in the recommendation list.

4.4. Parameter Settings. We set the latent vectors' dimensionality to $d = 100$ for the two datasets, like the settings of [20, 21, 37]. Besides, all parameters are set initially by a Gaussian distribution $N(0, 0.1^2)$, where its mean is 0 and $1/\sqrt{d}$ is its standard deviation. The initial learning rate is set to 0.001 and the batch size is set to 100. Recall that we use Adam optimizer [36] to update all parameters. For NARM, we set the mini-batch size to 512, the learning rate to 0.001, and the epochs to 30, respectively. For STAMP, the learning rate decay is set to 1 and the latent vectors' dimensionality is set to 100. For SRGNN, we set the learning rate decay to 0.1 behind every 3 epochs and the dimensionality is also 100. These key variables are shown in Table 2.

Table 2: The key variables of RMMCNN.

Variables	Value
Dimension	100
Learning rate	0.001
Batch size	100
Hidden size	100
Output size	100
L2 penalty	10^{-5}

4.5. Experimental Results. In this section, we compare RMMCNN in various aspects. Firstly, we compare RMMCNN with state-of-the-art methods. Then, we compare the performance of RMMCNN methods with variants of session embeddings. Finally, we compare the performance of RMMCNN with different evaluation metrics.

4.5.1. Comparison with Baseline Methods. To compare the performance of our proposed model, we compare it with some baseline models: NARM, STAMP, and SRGNN. The results show that RMMCNN outperforms all of them. The overall output is given in Figures 2 and 3; further detailed overview of the results can be seen in Table 3. RMMCNN aggregates the external and internal features of the session into the final data and makes recommendations based on both data. Compared with the current mainstream recommendation methods based on neural networks, RMMCNN shows excellent performance. NARM captures the user's overall interest through a cyclic neural network, and STAMP adds the last clicked event to the information extraction of the recommended item to achieve the purpose of information enhancement. These neural network-based methods achieve better recommendation performance than other methods. Figure 2 shows that STAMP is very low on the Diginetica dataset; this may be due to the fact that STAMP only uses the transition between users' last click item and users' historical click item. This information may not be enough to predict user session behavior. Therefore, the performance of STAMP in Diginetica dataset is very poor. SRGNN [21] generally considers the sequence characteristics of the user's click items and the user's last click event, which also combines the global characteristics and the local characteristics of the last click event for recommendation.

Our proposed model, RMMCNN, considers not only the external characteristics but also the internal characteristics of the session. Therefore, it combines the sequential characteristics of the session, the internal item characteristics of the session, and the last clicked event of the user to make recommendations. Simultaneously, RMMCNN uses an attention mechanism to automatically learn the weight

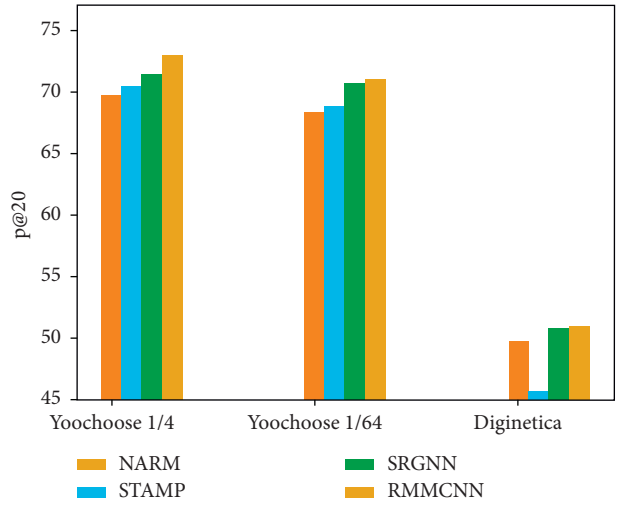

Figure 2: P@20 of different models.

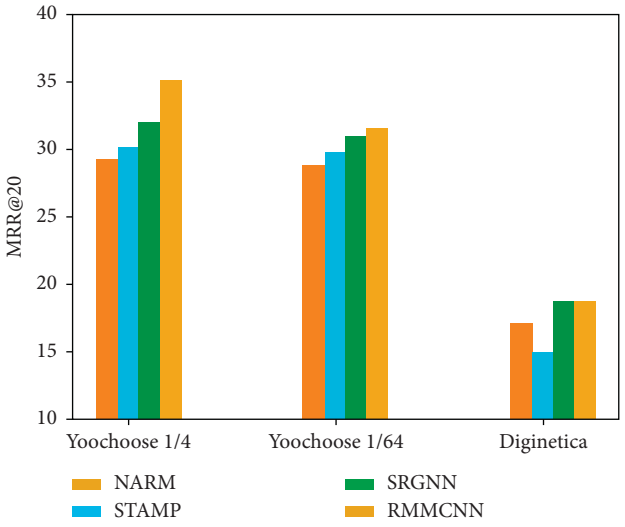

Figure 3: MRR@20 of different models.

Table 3: The performance of different datasets (K = 20).

Datasets	Yoochoose 1/4 (%)		Yoochoose 1/64 (%)		Diginetica (%)	
Measures	P@K	MRR@K	P@K	MRR@K	P@K	MRR@K
NARM	69.73	29.23	68.32	28.63	49.70	16.17
STAMP	70.44	30.00	68.74	29.67	45.64	14.32
SR-GNN	71.36	31.89	70.57	30.94	50.73	17.59
RMMCNN	72.93	35.04	70.94	31.46	51.82	18.19

representation. Thus, RMMCNN has obtained a richer user session representation and can make better recommendations for different user behaviors.

4.5.2. Comparison with Variants of Session Embeddings. We conduct distributed experiments on the model to verify the rationality of the RMMCNN model connection method

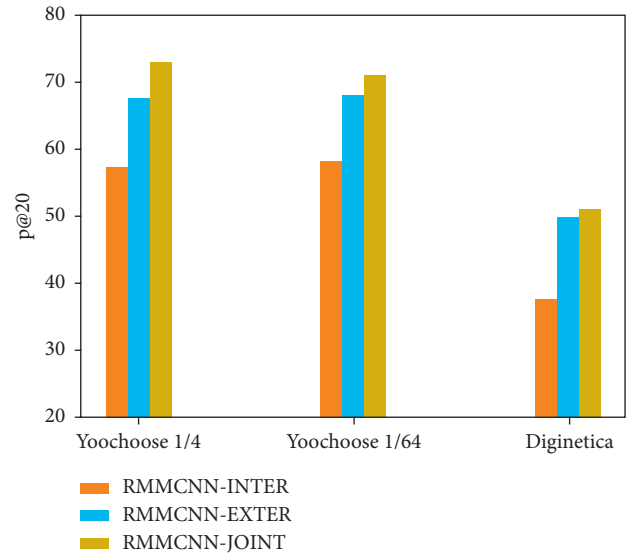

Figure 4: P@20 of different connection schemes.

and conduct experiments on the internal and external recommendation methods. We define the sequence feature of the session as an external feature and the relationship between users and items as an internal feature. We call the two methods RMMCNN-EXTER and RMMCNN-INTER, respectively.

(i) RMMCNN-EXTER: We first model the session as a directed graph and then extract the sequence features of the session through the GRU unit. Finally, we obtain different weight representations through the attention network.

(ii) RMMCNN-INTER: We obtain the embedding vector of the session to represent the internal relationship through a multichannel convolutional neural network. After that, we can get richer embedded vector information.

(iii) RMMCNN-JOINT: We combine the external vector with the internal vector of the session to generate the final session embedding representation.

Through Figures 4 and 5, we can see that the recommendation performance of RMMCNN-JOINT is better than that of the other two recommendation models. This is because RMMCNN-JOINT embeds more user information, and the content representation for recommendation is also richer. More details can be viewed in Table 4.

4.5.3. Comparison with Different Evaluation Metrics. In order to further measure the recommendation performance of different methods, we use different evaluation metrics to evaluate them. As illustrated in Figure 6, the results show that as the value of K increases, the recommendation accuracy rate and recommendation performance decrease, which indicates that the ideal recommendation result ranking position is not high. On the other hand, from the results, we can see that the recommendation performance

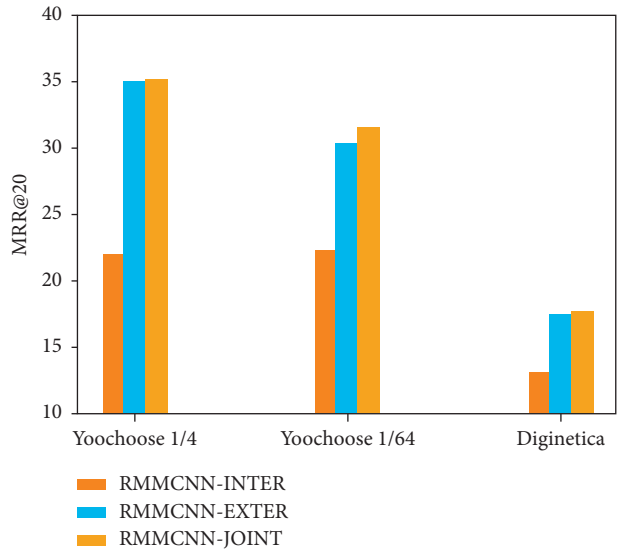

Figure 5: MRR@20 of different connection schemes.

Table 4: The performance of diverse session embeddings (K = 20).

Methods	Internal (%)		External (%)		Joint (%)	
Measures	P@K	MRR@K	P@K	MRR@K	P@K	MRR@K
Yoochoose 1/4	57.20	21.96	67.64	34.90	72.93	35.04
Yoochoose 1/64	58.10	22.22	67.87	30.26	70.94	31.46
Diginetica	37.56	12.97	49.71	17.38	51.82	18.19

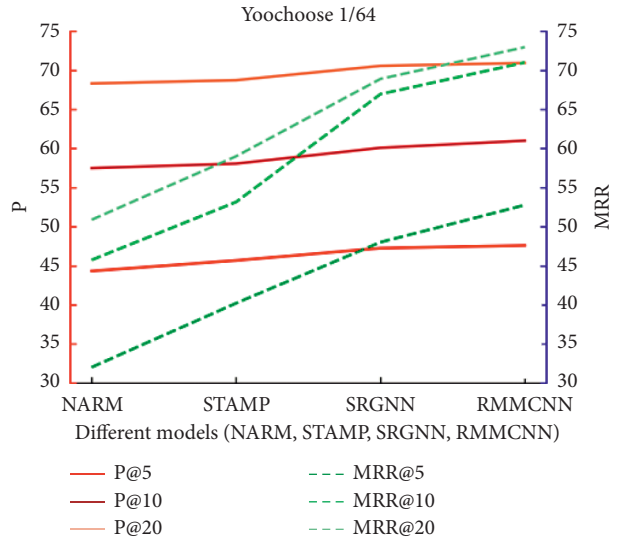

Figure 6: The Precision and MRR of different K settings on Yoochoose 1/64.

improves as the model complexity increases. This shows that as the complexity of the model increases, the embedded information extracted by the model is richer, and the recommendation results rank higher.

We tested on the Yoochoose dataset and Diginetica dataset. On the Yoochoose dataset, the result of P@10 is 60.32% and the result of P@20 is 70.94%. The results are better than those in other baseline models. On the Diginetica dataset, the result of P@10 is 36.64%, which is lower than 39.89% of NARM. However, the result of P@20 is 51.82%, while the result of SRGNN is 50.73%, indicating that the ranking of RMMCNN recommendation results in Diginetica data is mostly in [10, 20].

In the indicators of P@10 in Table 5, we see that NARM is superior to other methods. This is mainly due to the NARM's hybrid encoder attention mechanism. In the

Table 5: The performance of various K (P5 means P@K with $K = 5$, M5 means MRR@K with $K = 5$, etc.).

Datasets	Yoochoose 1/64						Diginetica					
Measures	P5	M5	P10	M10	P20	M20	P5	M5	P10	M10	P20	M20
NARM	44.34	26.21	57.50	27.97	68.32	28.63	27.95	15.67	**39.89**	17.26	49.70	16.17
STAMP	45.69	27.26	58.07	28.92	68.74	29.67	20.41	10.87	30.46	12.20	45.64	14.32
SR-GNN	47.26	28.26	60.09	30.69	70.57	30.94	26.64	15.10	37.91	16.61	50.73	17.59
RMMCNN	**47.61**	**28.87**	**60.32**	**30.71**	**70.94**	**31.46**	**26.83**	**15.90**	36.64	**17.93**	**51.82**	**18.19**

Diginetica dataset, the attention mechanism is more complex than those in other models. More representation information is aggregated, so the performance is significantly better than those of other recommendation models. However, as the K value increases, the recommendation performance of NARM is not as good as those of other more complex models. We argue that more complex models introduce more dimensional representation information, which is much better than simply using the attention mechanism.

5. Conclusions

In this paper, we first propose a novel multichannel convolution model to capture the rich information of items for recommendation. Then, we use the attention mechanism to obtain the features of the user clicked items sequence adaptively and combine the internal and external features of the session to jointly generate the final users' session vector embedding. Once we have the vector, we perform a linear transformation along with the softmax function. We convert the result into a probability value between $[0, 1]$. We have conducted some experimental results with stat-of-the-art recommender systems with two real datasets and both Precision and MRR have been improved. In the future, we will further study the richer representations of items' embedding to make more accurate recommendations.

Data Availability

The data used can be found at https://2015.recsyschallenge.com/challenge.html and https://competitions.codalab.org/competitions/11161#learn_the_details-data2.

Conflicts of Interest

There are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Major Project of the National Natural Science Foundation of China (no. 51935002) and the National Key Research and Development Program of China (no. 2018YFC0831903).

References

[1] Z. Y. Ji, H. Y. Pi, and W. Yao, "A hybrid recommendation model based on fusion of multi-source heterogeneous data," *Journal of Beijing University of Posts and Telecommunications*, vol. 42, no. 1, pp. 126–132, 2019.

[2] D. R. Liu, Y. C. Chou, and C. T. Jian, "Integrating collaborative topic modeling and diversity for movie recommendations during news browsing," *Kybernetes*, vol. 49, no. 11, pp. 2633–2649, 2019.

[3] Y. Hu, F. Xiong, D. Lu, X. Wang, X. Xiong, and H. Chen, "Movie collaborative filtering with multiplex implicit feedbacks," *Neurocomputing*, vol. 398, pp. 485–494, 2020.

[4] H.-Y. Chang, S.-C. Huang, and J.-H. Wu, "A personalized music recommendation system based on electroencephalography feedback," *Multimedia Tools and Applications*, vol. 76, no. 19, pp. 19523–19542, 2017.

[5] J. W. Chang, C. Y. Chiou, J. Y. Liao et al., "Music recommender using deep embedding-based features and behavior-based reinforcement learning," *Multimedia Tools and Applications*, Springer, Berlin, Germany, 2019.

[6] F. M. Belém, R. M. Silva, C. M. V. de Andrade et al., "Fixing the curse of the bad product descriptions"—search-boosted tag recommendation for E-commerce products," *Information Processing & Management*, vol. 57, no. 5, Article ID 102289, 2020.

[7] K. Wang, T. Zhang, T. Xue, Y. Lu, and S.-G. Na, "E-commerce personalized recommendation analysis by deeply-learned clustering," *Journal of Visual Communication and Image Representation*, vol. 71, Article ID 102735, 2020.

[8] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," *User Modeling and User-Adapted Interaction*, vol. 28, pp. 331–390, 2018.

[9] L. M. Zhang, P. Liu, and J. A. Gulla, "Dynamic attention-integrated neural network for session-based news recommendation," *Machine Learning*, vol. 108, pp. 1851–1875, 2019.

[10] M. Shi, J. X. Liu, and D. Zhou, "A hybrid approach for automatic mashup tag recommendation," *Journal of Web Engineering*, vol. 16, pp. 676–692, 2017.

[11] V. Maihami, D. Zandi, and K. Naderi, "Proposing a novel method for improving the performance of collaborative filtering systems regarding the priority of similar users," *Physica A: Statistical Mechanics and Its Applications*, vol. 536, Article ID 121021, 2019.

[12] C. Zou and Z. Chen, "Joint latent factors and attributes to discover interpretable preferences in recommendation," *Information Sciences*, vol. 505, pp. 498–512, 2019.

[13] J. Liu, L. Fu, X. Wang, F. Tang, and G. Chen, "Joint recommendations in multilayer mobile social networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 10, pp. 2358–2373, 2020.

[14] J. H. Wang and T. W. Liu, "Improving sentiment rating of movie review comments for recommendation," in *Proceedings of the IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW)*, pp. 12–14, Taipei, Taiwan, June 2017.

[15] Z. Y. Ji, X. J. Song, H. Y. Pi, and C. Yang, "Recommended model for fusing multi-source heterogeneous data based on

deep learning," *Journal of Beijing University of Posts and Telecom*, vol. 42, no. 6, pp. 35–42, 2019.

[16] W. Zhang, Z. Zhang, S. Zeadally et al., "MASM: a multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4216–4224, 2019.

[17] W. Zhang, Z. Zhang, H. C. Chao et al., "Toward intelligent network optimization in wireless networking: an auto-learning framework," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 76–82, 2019.

[18] J. Li, H. Xu, X. W. He et al., "Tweet modeling with LSTM recurrent neural networks for hashtag recommendation," in *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 24–29, Vancouver, Canada, July 2016.

[19] J. F. Dong, X. R. Li, C. X. Xu et al., "Feature re-learning with data augmentation for content-based video recommendation," in *Proceedings of the 26th ACM Multimedia Conference (MM)*, pp. 22–26, Seoul, South Korea, October 2018.

[20] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "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*, pp. 19–23, London, UK, August 2018.

[21] S. Wu, Y. Y. Tang, Y. Q. Zhu et al., "Session-based recommendation with graph neural networks," 2019, https://arxiv.org/abs/1811.00855.

[22] M. Jiang, Z. Y. Yang, and C. Zhao, "What to play next? A RNN-Based Music Recommendation System," in *Proceedings of the 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 356–358, Pacific Grove, CA, USA, November 2017.

[23] B. Xia, Y. Li, Q. M. Li et al., "Attention-based recurrent neural network for location recommendation," in *Proceedings of the 12th International Conference on Intelligent Systems and Knowledge Engineering (IEEE ISKE)*, Nanjing, CHINA, November 2017.

[24] Y. Cai, S. B. Dong, and J. L. Hu, "Jointly modeling user and item reviews by CNN for multi-domain recommendation," in *Proceedings of the 24th China conference on information retrieval (CCIR)*, September 2018.

[25] R. Gao, H. F. Xia, J. Li et al., "DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation," in *Proceedings of the 19th IEEE International Conference on Data Mining (ICDM)*, Beijing, China, November 2019.

[26] W. Q. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *Proceedings of the World Wide Web Conference (WWW)*, San Francisco, CA, USA, May 2019.

[27] X. Xian, L. Fang, and S. Sun, "ReGNN: a repeat aware graph neural network for session-based recommendations," *IEEE Access*, vol. 8, pp. 98518–98525, 2020.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the ACM World Wide Web Conference*, pp. 285–295, April 2001.

[29] W. Cheng, G. Yin, Y. Dong, H. Dong, and W. Zhang, "Collaborative filtering recommendation on users' interest sequences," *PLoS One*, vol. 5, Article ID e0155739, 2016.

[30] W. T. H. Putri, M. S. Prastio, R. Hendrowati et al., "Content-based filtering model for recommendation of Indonesian legal article study case of klinik hukumonline," in *Proceedings of the 4th International workshop on big data and information security (IWBIS)*, Bali, Indonesia, October 2019.

[31] H. Zhang, Y. Sun, M. Zhao, T. W. S. Chow, and Q. M. J. Wu, "Bridging user interest to item content for recommender systems: an optimization model," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4268–4280, 2020.

[32] T. Trinh, D. M. Wu, R. L. Wang et al., "An effective content-based event recommendation model," *Multimedia Tools and Applications*, Springer, Berlin, Germany, 2020.

[33] E. Rojsattarat and N. Soonthornphisaj, "Hybrid recommendation: combining content-based prediction and collaborative filtering," in *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning*, Hong Kong, China, March 2003.

[34] M. Kiewra, "RankFeed—recommendation as searching without queries: new hybrid method of recommendation," *Journal of Universal Computer Science*, vol. 11, no. 2, pp. 229–249, 2005.

[35] M. Kolahkaj, A. Harounabadi, A. Nikravanshalmani et al., "A hybrid context-aware approach for e-tourism package recommendation based on asymmetric similarity measurement and sequential pattern mining," *Electronic Commerce Research and Applications*, vol. 42, 2020.

[36] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, Banff, Canada, April 2014.

[37] J. Li, P. J. Ren, Z. M. Chen et al., "Neural attentive session-based recommendation," in *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, Singapore, Singapore, November 2017.