ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 78-87
doi: 10.18178/wcse.2019.06.013

Sequential Recommendation with Recurrent Convolutional Model

Shiyu Peng, Jiaxing Song+, Weidong Liu

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
psy16@mails.tsinghua.edu.cn, jxsong@tsinghua.edu.cn, liuwd@tsinghua.edu.cn

Abstract. Personalized sequential recommendation refers to making recommendation based on users' historical consumption behaviors. Most works based on RNN only model long-term patterns, which fail to capture skip behaviors. Contrarily, the CNN-based model whose target is to handle this problem can only leverage part of sequential behaviors and ignores global patterns, which limits its performance. In this paper, we propose a Recurrent Convolutional Recommendation Model (RCRM) to simultaneously catch global and local patterns. Specifically, we employ a recurrent layer to capture global patterns and a convolutional layer to extract local patterns. An attention mechanism is then introduced to generate the final attentive local pattern, which can further concatenate with global patterns to predict next item. We conduct extensive experiments on two benchmark datasets and the results demonstrate that RCRM outperforms state-of-the-art baselines by a large margin over a variety of common evaluation metrics.

Keywords: Sequential recommendation; recurrent neural networks; convolutional neural networks; attention mechanism

1. Introduction

In normal recommender systems, the consecutive events of one user are assumed to be independent and the dynamic preferences of the user are neglected. Actually, in many situations of real world, users' preferences are influenced by their previous behaviors. For instance, one may want to see The Avengers II and III after having watched The Avengers I. And users may continue to buy the same commercial goods for their satisfying experiences. The sequential recommendation is proposed to employ these historical behaviors.

For user sequential behaviors, there are always two kinds of patterns, global patterns and local patterns. While global patterns represent long-term dynamic preferences, the local patterns indicate important short-term interests which may skip a few time steps. Suppose a shopping example that one user buys a bicycle and basket first, then he continues to buy some snacks and drinks. After that, he buys a pump. In this case, if the recommender only considers global patterns, it would recommend the snacks and drinks with a high probability, because global patterns show that user's interests have already transferred to snacks and drinks. However, if the recommender takes local patterns into consideration, the pump can also be a competitive candidate to be recommended, since the recommender can find important previous short patterns.

Although global patterns and local patterns are both important to sequential recommendation, existing methods always consider only one of them. Recurrent neural network (RNN), as a specialized method to model sequence data, has become a popular choice in sequential recommendation. In [8], RNN is first introduced to session-based recommendation. Then [1,9,10] make improvements based on that. And [4] extends RNN for user-based recommendation. Even if RNN has inherent power of modeling sequences, it usually assumes temporal dependency changes monotonously along with time steps, which results in its incapability of seizing short-term patterns. From a long-term perspective, this property makes RNN fit global patterns properly, inasmuch as the recent items are actually more important than those items have been consumed

+ Corresponding author. Tel.: 13911567550
E-mail address: jxsong@tsinghua.edu.cn

long time ago. Yet when modeling local patterns, the impact of previous patterns may skip a few steps and still maintain strength. Due to this shortcoming of RNN, [16] proposes a convolutional sequence embedding recommendation model to capture the foregoing skip patterns. However, the model only takes several latest items in a fixed size window which ignores the global sequential patterns.

To address the limitations of previous work, we propose a Recurrent Convolutional Recommendation model to simultaneously model global and local patterns. First, relying on the intrinsic power of RNN to process sequences, recurrent layers are applied to capture global sequential patterns. Second, through convolutional layers equipped with gated linear units (GLU), we can model local patterns of continuous items within a fixed length window. These local patterns in different positions are automatically selected by a followed attention mechanism. Then the combined global sequential patterns and attentive local patterns are used to compute recommendation scores for each candidate item. We theoretically and empirically show that our model is capable to make up the shortage of RNN and CNN, and also make full use of their advantages. The results of the experiments on two real-world datasets indicate significant improvements compared with state-of-the-art methods.

2. Related Work

In this section, we first review some traditional methods in sequential recommendation, then we discuss deep learning based methods which are closely related to ours.

2.1 Traditional methods

One of the early solutions to sequential recommendation problem is Markov Chain. Based on the Markov assumption, a number of MC based models are proposed. [20] describes how to extract sequential patterns and builds a recommender based on Markov chain. In [15], Markov decision processes are adopted to model user behavior sequences and the maximum likelihood estimates are enhanced with some heuristic methods. Combining both advantages of MC and matrix factorization, factorized personalized MC (FPMC) model is proposed to predict next basket items [7]. Then FPMC is extended to embed the localized regions, which not only exploits the personalized Markov chain, but also considers users' movement constraint [14]. [12] studies different sequential patterns for recommendation and finds that contiguous sequential patterns are more suitable for sequential prediction task than general sequential patterns. However, MC-based models might face the problem of independence assumption among past components. Additionally, with applying MC to sequential recommendation, the state space quickly becomes unmanageable when models include all possible sequences.

2.2 Deep learning based methods

Deep learning is an effective representation learning technique, which has been applied quite successfully in a number of areas. As one of these methods designed for learning temporal aspects, RNN is widely used in sequence modeling tasks. For sequential recommendation, RNN is also a popular choice. [8] first applies RNN in session-based recommendation and proposes GRU4Rec model, which achieves significant improvements over conventional methods. Based on GRU4Rec, [1] explores the value of incorporating dwell time. The general idea of [1] is that the longer user examines an item, the more interested s/he is in that item. [9] introduces several parallel RNN architectures to model clicks and the features of the clicked items, such as images and texts. In [4], a delicately designed GRU with user integration replaces standard GRU to make more personalized recommendations. Although the above RNN based models have achieved encouraging results, they still suffer from the problem that RNN models sequence that one element usually has more significant effect than the previous one for prediction.

Besides RNN, Convolutional Neural Network (CNN) is another common method to model sequences, especially in natural language processing. Yet in recommender system, CNN is not used broadly. It is basically applied to process images and texts for recommendation. In [19], for instance, convolutional layers are introduced to extract users' features from their reviews. And [18] leverages CNN to learn movie content from the poster. In sequential recommendation, [16] proposes a pioneering CNN model called Caser to make personalized recommendation. Its idea is to embed sequence of recent items into an "image" along the time and learn sequential patterns as local features with convolutional filters. However, modeling sequences with fixed size, Caser does not consider long-term sequential preference, which limits the performance.

3. Proposed Methodology

In this section, we first introduce the general architecture of RCRM model. Then we describe each component of the model in detail respectively.

3.1 Overview

In this paper, we propose a novel CNN and RNN combined model to address personalized sequential recommendation problem, named Recurrent Convolutional Recommendation Model (RCRM). The basic idea of RCRM is to model both global and local patterns, and then make personalized recommendation based on that. As shown in Figure 1, the model consists of three main components, dynamic generator (including global learner and local learner), static generator, and recommendation generator.

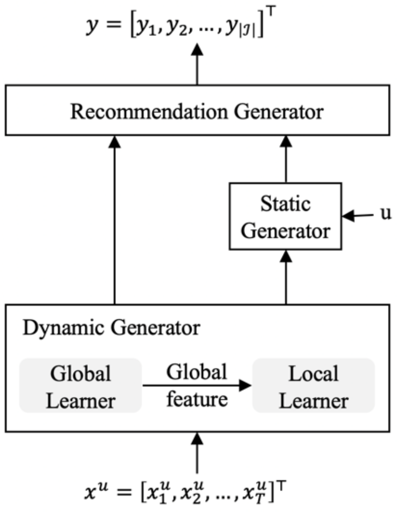

Fig. 1: The general architecture of RCRM.

Dynamic Generator takes item sequence $\mathrm{x^u} = [\mathrm{x_1^u}, \mathrm{x_2^u}, \dots, \mathrm{x_t^u}]^\top$ as input, where $\mathrm{x_i^u} \in \mathrm{I}, 1 \leq \mathrm{i} \leq \mathrm{t}$ and outputs user's dynamic interests with global learner and local learner. The role of global learner is to learn long-term sequential patterns with RNN. Although RNN can model global patterns, it hardly makes full use of most important local sequential patterns, for the reason that RNN assumes dependency changes monotonously along time. The local learner is built to make up for this. With one-dimensional convolution, local learner captures local patterns within a fix-sized local window. Taking global sequential behavior into consideration, local learner computes attention for each step. The global information and attentive local information are then used to produce dynamic preference. Static generator converts the input user u into high dimensional user embedding to model static preference. Finally, dynamic preference and static preference are fed to the recommendation generator to generate a ranking list over all items, $\mathrm{y} = [\mathrm{y_1}, \mathrm{y_2}, \dots, \mathrm{y_{|I|}}]^\top$, that can be the next item user going to consume.

The main motivation of our work is that RNN and CNN are two models which can complement each other. Compared to recurrent networks, convolutional networks don't have dependency changing monotonously problem and are specialized to extract local patterns that are highly correlated. But convolutions create representations for fixed size sequences. To increase effective sequence size, the network is going to get deeper, which may bring new problems [6]. Therefore, RNN is adopted to learn global patterns for the reason of its capacity to model long-term sequences.

3.2 Global Learner

The global learner, shown in the Figure 2, is a RNN with Gated Recurrent Unit (GRU) [2]. GRU is a more elaborate RNN unit aiming to deal with vanishing gradient problem. Each GRU has two gates, an update gate and a reset gate. The update gate determines how much of the previous memory to keep,

$$z_t = \sigma(W_z e_t + U_z h_{t-1}). \tag{1}$$

The reset gate determines how to combine the new input with the previous memory,

$$r_t = \sigma(W_r e_t + U_r h_{t-1}). \tag{2}$$

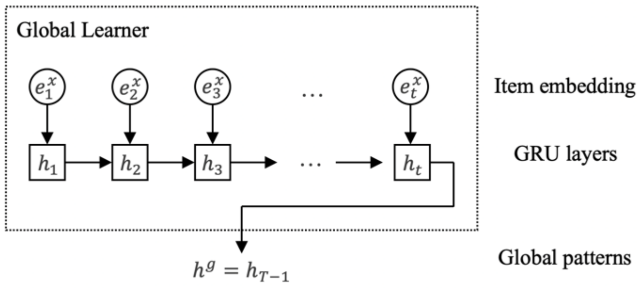

Fig. 2: The global learner detached from the complete model.

Then the candidate activation $\widehat{h_t}$ is computed in a similar manner,

$$\widehat{h_t} = \tanh(We_t + U(r_t \odot h_{t-1})). \tag{3}$$

Finally, the activation of the GRU is a linear interpolation between the previous activation and the candidate activation,

$$h_t = (1 - z_t)h_{t-1} + z_t\widehat{h_t}. \tag{4}$$

Because the final hidden state h_t captures information in all the previous time steps, we take $h_t \in \mathbb{R}^d$ as the expected global patterns $h^g \in \mathbb{R}^d$:

$$h^g = h_t. \tag{5}$$

3.3 Local Learner

Local learner leverages the recent success of convolution filters of CNN on sequence learning [5,11]. With a shallow convolutional network, the learner could capture local sequential patterns. The inputs of local learner are a sequence of item embedding $e^x \in \mathbb{R}^{t\times d}$ which are the same as global learner, while the outputs are attentive local patterns $c^l \in \mathbb{R}^d$.

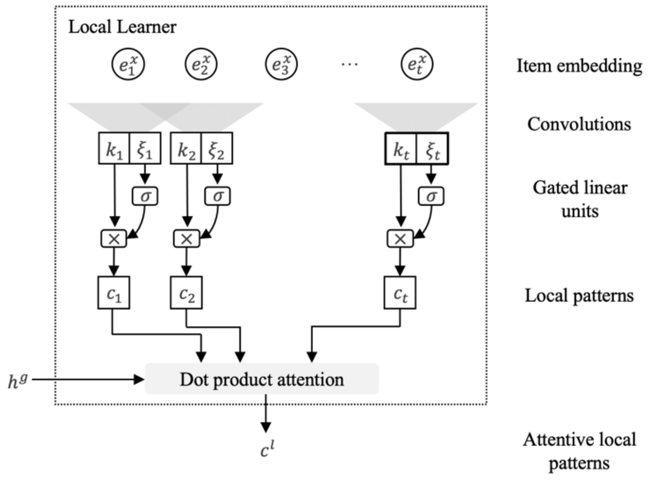

Fig. 3: The local learner detached from the complete model.

Figure 3 shows the detailed structure of local learner. The one-dimensional convolution has *2d* filters $f_j \in \mathbb{R}^{wd}$, where $1 \le j \le 2d$ and $w \in \{1, \dots, t\}$. To capture features for total *n* steps, we ensure the length of outputs after convolution is the same as inputs by padding. Each f_j slides from beginning to the end of the sequence. The result of the interaction for the i^{th} convolution value is given by,

$$q_{ij} = [e_i^x\ e_{i+1}^x \cdots e_{i+w-1}^x] \odot f_j + b_j^f, \tag{6}$$

where the symbol $\odot$ denotes the inner product operator, *i* denotes the i^{th} time step and *j* denote j^{th} the filter. The result $q_{ij} \in \mathbb{R}$ is the inner product between filter f_j and items within w-sized local window, it contains sequential patterns among these items. As there are altogether *2d* filters, the output of the convolution for one time step is $q_i \in \mathbb{R}^{2d}$. However, for each time step, not all patterns are equally important for the current prediction. We therefore take GLU [3] as non-linearity and attach weight to the patterns. The convolutional output $q_i = \begin{bmatrix} k_i \\ \xi_i \end{bmatrix}$ is the input of GLU:

$$c_i = k_i \otimes \sigma(\xi_i). \tag{7}$$

where the symbol $\otimes$ is point-wise multiplication, σ represents sigmoid function, $k_i, \xi_i \in \mathbb{R}^d$ and $c_i \in \mathbb{R}^d$ is half the size of q_i. The gate $\sigma(\xi_i)$ dynamically controls the relevant parts of q_i that are exploited to produce the final local sequential patterns. For there are t time steps, the local patterns can be represented as $C = [c_1\ c_2 \dots c_t] \in \mathbb{R}^{t \times d}$.

As discussed before, local learner is designed to capture relevant local patterns. One common technique to extract important elements in CNNs is max pooling. However, the relevant elements are also influenced by user's global intention. Thus we involve a dot-product attention [17] which allows the model to automatically learn attention for every time step with regard to global patterns,

$$\alpha_i = \frac{exp(h^g \odot c_i)}{\sum_{j=1}^{t} exp(h^g \odot c_j)}, \tag{8}$$

where α_i is the weight of patterns at time step i. Then different elements of local patterns are combined linearly as follows:

$$c^l = \sum_{i=1}^{t} \alpha_i c_i, \tag{9}$$

where $c^l \in \mathbb{R}^d$ represents the final attentive local patterns.

This local learner enjoys the advantages of adaptively focusing on significant parts of local patterns by two mechanisms: First, the learner only selects important patterns within local window by GLU. Second, dot-product attention mechanism computes the weights for different time steps.

3.4 Recommendation

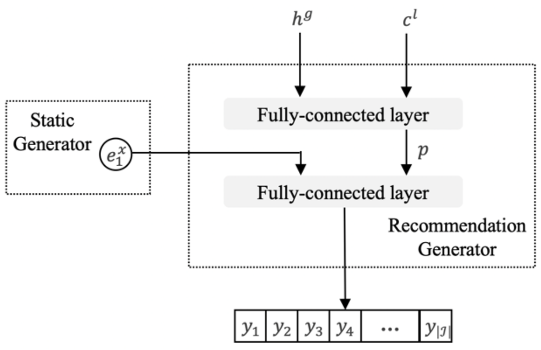

Fig. 4: The static generator and recommendation generator detached from the complete model.

The global learner and local learner play different roles in our model. The former learns global patterns of the whole sequence, while the latter learns attentive local patterns, both of which are useful for predicting next item. Naturally we combine the global patterns and local patterns and feed them into a fully-connected layer to get an extended dynamic preference representation $p \in \mathbb{R}^d$,

$$p = \phi(W^p \begin{bmatrix} h^g \\ c^l \end{bmatrix} + b^p), \tag{10}$$

where $W^p \in \mathbb{R}^{d \times 2d}$ is the weight matrix that projects the concatenation to a d-dimensional vector, $b^p \in \mathbb{R}^d$ denotes the corresponding bias term and $\phi(\cdot)$ represents the activation function.

As shown in Figure 4, to model user's static preference, we look up the user embedding e^u in static generator. The two d-dimensional vectors p and e^u are also concatenated together and enter into a fully-connected layer:

$$y = W^o \begin{bmatrix} p \\ e^u \end{bmatrix} + b^o, \tag{11}$$

where $W^o \in \mathbb{R}^{|I| \times d}$ and $b \in \mathbb{R}^{|I|}$ are the weight matrix and bias term respectively. Each element of the results $y = [y_1, y_2, \dots, y_{|I|}]^{\mathrm{T}} \in \mathbb{R}^{|I|}$ represents the likelihood that corresponding item will occur next.

3.5 Network Training

In the training process, the output of the final layer is entered to a sigmoid function:

$$p(\mathcal{C}_t^u | x_1^u, x_2^u, \dots, x_t^u) = \sigma\left(y_{\mathcal{C}_t^u}^u\right), \tag{12}$$

where $\mathcal{C}_t^u$ is the collection of candidate items for the user u at the time step n.

The collection $\mathcal{C}_t^u$ is the union of two collections, the positive collection $\mathcal{P}_t^u$, and negative collection $\mathcal{N}_t^u$. As our goal is to predict the next item a target user u is going to consume, the positive collection $\mathcal{P}_t^u$ always contains the next item x_{t+1}^u. Following the previous work [16], we randomly sample several negative items for each positive item to form collection $\mathcal{N}_t^u$.

For the whole item sequence a user u has consumed in the training set $x^u = [x_1^u, x_2^u, \dots, x_{n-1}^u, x_n^u]$, we generate sequences and corresponding labels $\{([x_1^u], x_2^u), ([x_1^u, x_2^u], x_3^u), \dots, ([x_1^u, \dots, x_{n-1}^u], x_n^u)\}$ for training. Let $\mathcal{T}^u$ be the collection of time steps we generation for the user u. Then the likelihood of all examples is,

$$\mathrm{P}(\mathcal{C}|\Theta) = \prod_u \prod_{t \in \mathcal{T}^u} \prod_{i \in \mathcal{P}_t^u} \sigma(y_i^u) \prod_{j \in \mathcal{N}_t^u} \left(1 - \sigma(y_j^u)\right). \tag{13}$$

Taking the negative logarithm of likelihood, the objective function, which is called binary cross entropy loss, is as follows:

$$\ell = \sum_u \sum_{t \in \mathcal{T}^u} \sum_{i \in \mathcal{P}_t^u} - \log\left(\sigma(y_i^u)\right) + \sum_{j \in \mathcal{N}_t^u} - \log\left(1 - \sigma(y_j^u)\right). \tag{14}$$

The model parameters $\Theta = \{W_z, U_z, W_r, U_r, W, U, f, b^f, W^p, b^p, W^o, b^o\}$ are learned by minimizing the objective function in equation (14) and the hyper-parameters are chosen via grid search. We take Adaptive Moment Estimation (Adam) as our optimization algorithms. To avoid overfitting, L2 Norm and dropout are adopted as regularization methods.

4. Experiments

4.1 Datasets and Experimental Setup

We evaluate our approach as well as all the baselines on two real-world datasets: MovieLens 1M[1] and LastFM[2]. For MovieLens 1M, we remove all rating scores to mimic implicit data. For LastFM, due to computational reasons, we perform our evaluation on a 10% subsample with a maximum length of 3000. To provide sequential recommendations, those items and users with less than 5 feedbacks are filtered out. The statistics of the final datasets are shown in Table 1.

Table 1: Datasets statistics.

Dataset	#records	#users	#items	avg.length
MovieLens 1M	998786	6064	3305	165.36
LastFM	896396	961	44060	932.77

Finally, we take first 70% of each user consumption sequence to form training set, the next 10% to form validation set, and the last 20% to form the test set. The hyper-parameters are tuned on the validation set while the final results are evaluated on the distinct test set after training is finished completely.

4.2 Baselines and evaluation metrics

We compare our proposed RCRM with four traditional methods (i.e., POP, Item-KNN, BPR, and FPMC) and a deep learning based method (i.e., Caser). POP always recommends the most popular items in training set, which is a strong baseline in certain domains. BPR [13] is a common method for non-sequential item recommendation on implicit feedback. FPMC [14] and Item-KNN are state-of-the-art methods for sequential recommendation. To show the advantages of our approach with consideration of both global and local se-

[1] http://grouplens.org/datasets/movielens/1m/

[2] https://www.last.fm/

quential patterns, we also compare it with Caser proposed in [16], which regards each embedded item sequence as an “image” and captures local sequential patterns on this “image” via CNN.

The goal of our approach is to predict the item that user would consume at a particular time step. Therefore, we recommend a few items at each time, among which the target item should be. To evaluate our model’s performance, we adopt four widely used metrics, Recall, Precision, F1-score and MRR.

4.3 Comparison against Baselines

Table 2 presents the results of all methods over two datasets, and the best performer on each column is highlighted in bold face. The last row is the improvement of RCRM relative to the best baseline, defined by $\frac{RCRM-baseline}{baseline}$.

Table 2: Performance comparison of RCRM with baselines over two datasets.

Dataset	MovieLens 1M				LastFM			
Measures@10	Precision	Recall	F1-score	MRR	Precision	Recall	F1-score	MRR
POP	0.0035	0.0349	0.0063	0.0118	0.0007	0.0067	0.0012	0.0029
BPR	0.0014	0.0145	0.0026	0.0046	0.0016	0.0159	0.0029	0.0057
Item-KNN	0.0143	0.1430	0.0260	0.0578	0.0157	0.1567	0.0285	0.0596
FMPC	0.0133	0.1325	0.0241	0.0451	0.0057	0.0572	0.0104	0.0269
Caser	0.0204	0.2036	0.0370	0.0740	0.01442	0.1442	0.0262	0.0727
RCRM	**0.0225**	**0.2255**	**0.0410**	**0.0832**	**0.0167**	**0.1670**	**0.0304**	**0.0972**
Improv.	**10.3%**	**10.8%**	**10.8%**	**12.4%**	**6.3%**	**6.6%**	**6.7%**	**33.7%**

As shown in Table 2, the sequential recommendation methods (i.e., Item-KNN, FMPC, Caser and RCRM) always outperform normal recommendation methods (i.e., POP and BPR), which indicates the importance of considering sequential information. Among all the methods, the proposed RCRM model performs best w.r.t. all metrics. This confirms that it is vital to model both global and local sequential patterns and RCRM has the capacity to exploit these two kinds of information.

On MovieLens 1M dataset, RCRM achieves a large improvement relative to the best baseline (i.e., Caser). One obvious reason is that RCRM takes users’ global dynamic preferences into consideration, which are important to long-term behavior of watching movies. And on LastFM dataset, except for MRR metric, the relative performance gain is lower than that on the MovieLens 1M. Given the nature of the datasets, this result is reasonable. When listening to music, users prefer to repeat the same song over and over again. In addition, they always consume a playlist or a whole album at a time. These behaviors lead to the result that recent sequential patterns have more effects than previous sequential patterns and the methods which only consider latest sequential information (e.g. Item-KNN and Caser) may have better performances on LastFM than on MovieLens 1M.

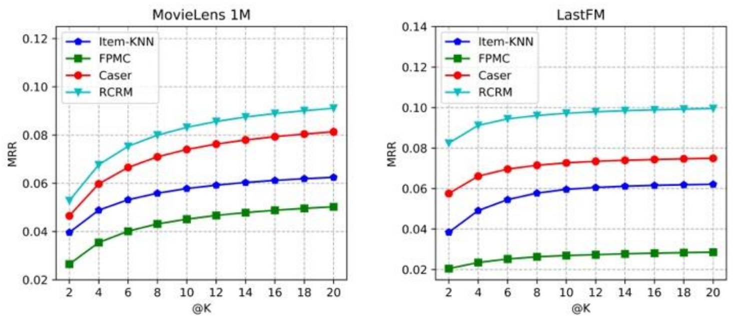

Fig. 5: Performance comparison between RCRM and strong baselines over two datasets with different recommendation list length K.

Meanwhile, Figure 5 demonstrates the performance comparison between RCRM and three strong baselines (i.e., Item-KNN, FPMC and Caser) with different recommendation list length K. On both two datasets, RCRM outperforms other methods stably, which proves that RCRM can offer high-quality recommendations.

Finally, concerning the baselines, BPR does not work well on MovieLens 1M, even worse than POP. For one thing, normal methods are no longer suitable for sequential recommendation. For another, people are more likely to watch popular movies. Although Caser outperforms Item-KNN with all metrics on MovieLens 1M, it performs worse than Item-KNN on LastFM dataset w.r.t Precision@10, Recall@10 and F_1@10. The main reason is the same as we explain before, that there are a number of repeated songs in users' consumption sequences owing to single cycle. In such situations, Item-KNN has more advantages to predict the next song.

4.4 Analysis of RCRM local learner

In this part, we evaluate the contribution of local learner. $\text{RCRM}_{\text{global}}$ refers to the RCRM that only uses global patterns by removing local learner. $\text{RCRM}_{\text{hybrid}}$ represents that the RCRM incorporates both local patterns and global patterns.

Table 3. Comparison among different versions of RCRM

Dataset	MovieLens 1M		LastFM	
Measures@20	F1-score	MRR	F1-score	MRR
$\text{RCRM}_{\text{global}}$	0.0291	0.0810	0.0183	0.0937
$\text{RCRM}_{\text{hybrid}}$	0.0326	0.0920	0.0191	0.0995

As show in Table 3, $\text{RCRM}_{\text{global}}$ do not perform well on two datasets. This indicates that merely considering long-term patterns may not be able to train a satisfying recommender. Besides, comparing the results on two datasets, local learner brings improvement with different degree. Specifically, $\text{RCRM}_{\text{hybrid}}$ performs better than $\text{RCRM}_{\text{global}}$ about *12.03%* and *13.58%* in terms of F1-score@20 and MRR@20 on MovieLens while *4.4%* and *6.2%* on LastFM. This verifies our assumptions again that on LastFM, recent information is more significant, where RNN can capture short-term patterns better compared with RNN in other cases. Therefore, it is understandable that local learner does not make contribution as much as on MovieLens.

4.5 Visualization of the attention weights

To illustrate the role of attention, we present some examples in Figure 6, which are sampled randomly from MovieLens. Each row corresponds to the 10 latest items a user has consumed, and the depth of the color shows the attention scores given by equation (8).

We have following observations from the figure: (1) The importance of historical items is not always increasing along the time, e.g., in example 3, the weight of item 5 is larger than the following items. Thus in one prediction, valuable information can appear any position in the sequence, which is one of the main motivation of RCRM. Recall that RNN assumes items importance grows over time monotonously, this also explains why hybrid model can outperform global model based only on RNN. (2) In some cases, one item plays a major role while in other cases, several items determine the next together. The effective items may be several steps apart or appear continuously. This further confirms the necessity of attention mechanism which highlights key information regardless of various forms of local patterns.

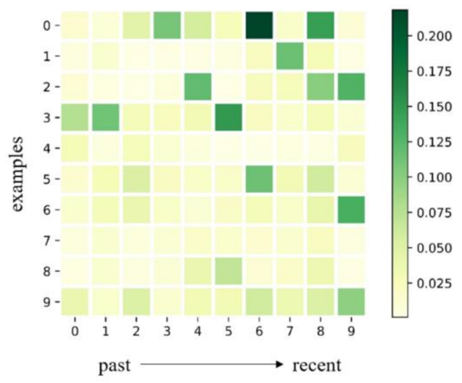

Fig. 6: Visualization of attentions for the last 10 items each user has interacted with.

5. Conclusion and Future Work

In this paper, aiming to sequential recommendation, we propose a novel model combining the recurrent layers and convolutional layers to capture both global patterns and local patterns. To further determine the importance of each item, the model integrates an attention mechanism. Based on global patterns and weighted local patterns, we have applied RCRM to predict next item the user would like to pick. The experiments have been conducted on two real-world datasets, and the results indicate that RCRM outperforms state-of-the-art methods in terms of four different metrics. And the analysis of local learner and attention mechanism verifies the effectiveness of our model.

At present, global sequential patterns and local sequential patterns are combined by simple concatenating. However, as we can see from the results, global patterns and local patterns are not equally important on different datasets. For the future work, we can adopt a more sophisticated structure to adaptively shift focus between these two patterns. Furthermore, other item features, such as film posters, ratings and reviews, can be integrated to enhance the model.

6. References

[1] Bogina, V. and Kuflik, T. 2017. Incorporating dwell time in session-based recommendations with recurrent Neural networks. In *CEUR Workshop Proceedings*, 57-59.

[2] Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y.J. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[3] Dauphin, Y.N., Fan, A., Auli, M. and Grangier, D. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

[4] Donkers, T., Loepp, B. and Ziegler, J. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* ACM, 152-160.

[5] Gehring, J., Auli, M., Grangier, D., Yarats, D. and Dauphin, Y.N. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

[6] He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

[7] He, R. and McAuley, J. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* IEEE, 191-200.

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

[9] Hidasi, B., Quadrana, M., Karatzoglou, A. and Tikk, D. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* ACM, 241-248.

[10] Jannach, D. and Ludewig, M. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* ACM, 306-310.

[11] Lai, S., Xu, L., Liu, K. and Zhao, J. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *AAAI*, 2267-2273.

[12] Mobasher, B., Dai, H., Luo, T. and Nakagawa, M. 2002. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on* IEEE, 669-672.

[13] Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* AUAI Press, 452-461.

[14] Rendle, S., Freudenthaler, C. and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web* ACM, 811-820.

[15] Shani, G., Heckerman, D. and Brafman, R. 2005. An MDP-based recommender system. *Journal of Machine Learning Research*, 1265-1295

[16] Tang, J. and Wang, K. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* ACM, 565-573.

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998-6008.

[18] Zhao, W., Wang, W., Ye, J., Gao, Y., Yang, M., Zhao, Z. and Chen, X. 2017. Leveraging Long and Short-term Information in Content-aware Movie Recommendation. *arXiv preprint arXiv:1712.09059.*

[19] Zheng, L., Noroozi, V. and Yu, P.S. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* ACM, 425-434.

[20] Zimdars, A., Chickering, D.M. and Meek, C. 2001. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence* Morgan Kaufmann Publishers Inc., 580-588.