Long Short-Term Memory with Sequence Completion for Cross-Domain Sequential Recommendation

Guang Yang[1](✉), Xiaoguang Hong[1](✉), Zhaohui Peng[1](✉), and Yang Xu[2](✉)

[1] School of Computer Science and Technology, Shandong University, Jinan, China
loggyt@yeah.net, {hxg,pzh}@sdu.edu.cn
[2] Shandong Normal University, Jinan, China
zzmylq@gmail.com

Abstract. As the emerging topic to solve the loss of time dimension information, sequential recommender systems (SRSs) has attracted increasing attention in recent years. Although SRSs can model the sequential user behaviors, the interactions between users and items, and the evolution of users' preferences and item popularity over time, the challenging issues of data sparsity and cold start are beyond our control. The conventional solutions based on cross-domain recommendation aims to matrix completion by means of transferring explicit or implicit feedback from the auxiliary domain to the target domain. But most existing transfer methods can't deal with temporal information. In this paper, we propose a Long Short-Term Memory with Sequence Completion (SCLSTM) model for cross-domain sequential recommendation. We first construct the sequence and supplement it in which two methods are proposed. The first method is to use the intrinsic features of users and items and the temporal features of user behaviors to establish similarity measure for sequence completion. Another method is to improve LSTM by building the connection between the output layer and the input layer of the next time step. Then we use LSTM to complete sequential recommendation. Experimental results on two real datasets extracted from Amazon transaction data demonstrate the superiority of our proposed models against other state-of-the-art methods.

Keywords: Cross-domain sequential recommendation · Long short-term memory · Sequence completion

1 Introduction

With the explosively growing amount of online information, recommender system (RS) is playing an indispensable role in our daily lives as well as in the Internet industry for the problem of information overload. The traditional RSs [1], including the content-based, collaborative filtering and hybrid collaborative filtering RSs, model the user-item interactions in a static way and lost the time dimension.

In the real world, users' shopping behaviors usually happen successively in a sequence, rather than in an isolated manner. Taking the real events of someone U_1

depicted in Fig. 1 as an example, U_1 bought a bag of infant formula milk, a baby stroller and diapers successively. So we can all guess about the likelihood of buying baby bottles. Likewise, the sequential dependencies can be seen in next case. Before U_2 started a vacation, he booked a flight, several tickets for some tourist attractions and rented a car successively, and his next action may be booking a local hotel. In such a case, based on the location of each attraction and car rental company, we can guess the location of the hotel. In the above scenario, each of U_2's next actions depends on the prior ones and therefore all the four consumer behaviors are sequentially dependent. Such kind of sequential dependencies commonly exist in actual data but cannot be well captured by the conventional collaborative filtering RSs or content-based RSs [2], which essentially motivates the development of sequential RSs.

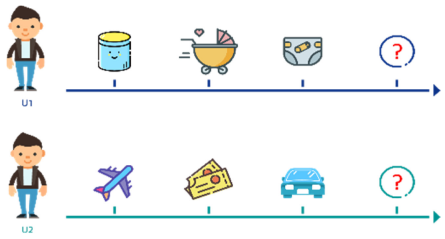

Fig. 1. Two examples of sequential RSs

Furthermore, user interest is dynamic rather than static over time [30]. How to capture user interest accurately to enhance the accuracy of recommendation results is an enormous practical challenge in RSs. For example, many people want to watch horror movies when Halloween comes and love movies are always popular on Valentine's Day. Such dynamics are of great significance for precisely profiling a user or an item for more accurate recommendations. The traditional RSs can't capture the dynamic change of interest or behavior well when sequential recommender systems (SRSs) are competent for the task.

In conclusion, sequential recommender systems meet our requirements for these objective situations, so they can greatly improve recommendation performance [3].

Unfortunately, recommender systems are generally faced with data sparsity and cold start problems in that users interact with only an extremely tiny part of the commodities on a website or a directory and sequential recommender systems are no exception. As a promising solution to address these issues, cross-domain recommender systems [4, 5] have gained increasing attention in recent years. This kind of algorithm tries to utilize explicit or implicit feedbacks from multiple auxiliary domains to improve the recommendation performance in the target domain. As shown in Fig. 2, commodities are divided into different domains according to their attributes or categories. Users who have active data in both domains are called linked users and we mark them with dashed red box. Users who only have active data in the target domain are called cold start users. Our goal is to make recommendations for cold start users by linked users. Linked users serve as a bridge for our model to transfer knowledge across domains. Existing studies [4, 5, 6, and 7] can't process sequence data. The sequence is split into separate data to fill in the user/item matrix and all temporal information is discarded. Therefore, how to deal with sequence data for cross-domain recommender systems remain an open problem.

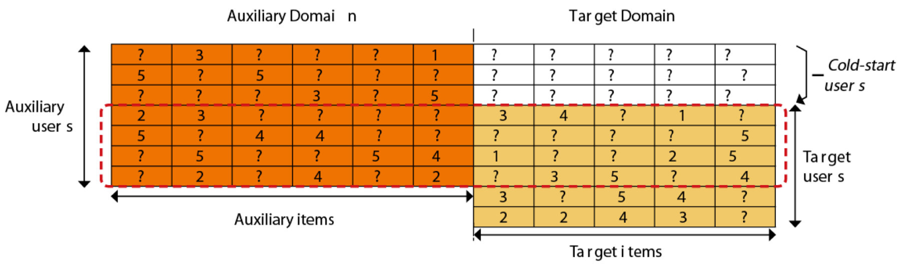

Fig. 2. Illustration of the cross-domain recommendation for cold-start users

To address the above challenges, we propose a Long Short-Term Memory with Sequence Completion (SCLSTM) model for cross-domain sequential recommendation in this paper. Specifically, we first construct the sequence and supplement it in which two methods are proposed. The first method is to use the intrinsic features of users and items and the temporal features of user behaviors to establish similarity measure for sequence completion. Another method is to improve Long Short-Term Memory network (LSTM) by building the connection between the output layer and the input layer of the next time step. Complete the sequence by adding this input logic unit. Then we use LSTM to complete sequential recommendation. Our major contributions are summarized as follows:

- We define the concept of sequence completion and propose two methods of sequence completion. One uses similarity measure, the other uses improved LSTM.

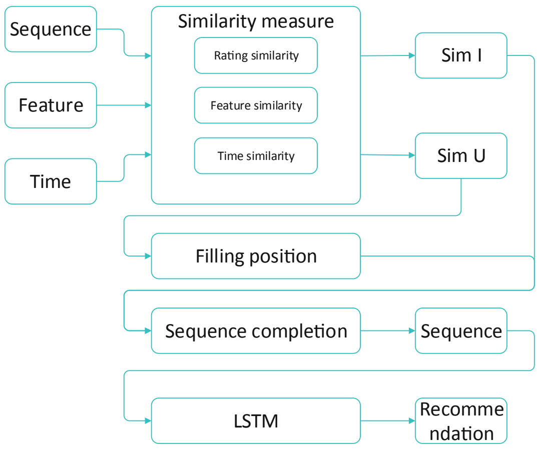

Fig. 3. SCLSTM with similarity measure recommendation framework

- We propose a Long Short-Term Memory with Sequence Completion (SCLSTM) model for cross-domain sequential recommendation, which can solve data sparsity and cold start problems of sequential recommender systems.
- We systematically evaluate our proposal through comparing it with the state-of-the-art algorithms on the dataset of Amazon[1]. The results confirm that our new method substantially improves the recommendation performance.

2 Sequence Completion

In real life, there are many sequences, such as a piece of text, a pulse signal and a melody. When we read a damaged book, it is difficult for us to understand it accurately. We must fill these sequences with external knowledge to solve these problems. For example, look at a fragmentary passage, "The Spider-Man series broke ground by featuring, a from behind Spider-Man's secret identity." Almost all superhero fans understand what it means. "The Spider-Man series broke ground by featuring Peter Parker, a high school student from Queens behind Spider-Man's secret identity." Why has this happened? These superhero fans use their animation knowledge to picture the sentence. We formally define the sequence completion as follows.

Definition 1. (Sequence Completion) For a sequence S_m of missing elements, the missing elements are recovered by analyzing the effective elements and context knowledge. Make the sequence S_m close to the complete sequence $\mathcal{S}$. ($\min\{|\mathcal{S} - S_m|\}$).

Take a word recognition as an example, given a word "spi_er" with a missing letter, we guess that the word is "spinner", "spider" or others according to dictionary information. Then, according to the context information, the word "spider" is determined. But when the word with a missing letter becomes "s_er", it's very difficult to determine what it really means. Common algorithms, especially RNN (Recurrent Neural Network)-based algorithms, can use sequence data to classify and are actually sequence completion algorithms. But these algorithms face two difficult problems. First of all, the sparser the data, the worse the algorithm performance. Especially the cold start problem is a fatal blow to these supervised algorithms. Secondly, the sequence of recommender system is quite different from the common sequence. Limited by the sampling method and platform, we cannot capture all the sequence data. Take book purchase record of Amazon as an example, the channels for users to obtain books include hypostatic stores, websites and libraries, even could borrow them from friends. But we can only collect part of the data through Amazon, so we can't get the real complete sequence on reading records.

For these reasons and more, we propose two novel algorithms of sequence completion.

3 SCLSTM with Similarity Measure

We propose a recommendation framework called SCLSTM based on similarity measure model and the framework is as shown in Fig. 3. First, we extract three kinds of data from

[1] https://www.amazon.com.

the logs, which they are users-items interaction sequences, users/items feature graph and users action time list. Then, we use these three groups of data to build three similarity measure models for users and items. The rating similarity, feature similarity and time similarity are calculated respectively, and combining three models to get user similarity SimU and item similarity SimI. Next, the user similarity is used to determine the location and length of sequence completion and item similarity is used to determine the content of sequence completion. Based on the previous step, we complete sequence completion and get relatively complete sequence data. Finally, we use the sequence we just obtained as input and use the LSTM model to recommend.

3.1 User and Item Similarity Measures

Rating Similarity Measure
We decide to use the classic cosine similarity algorithm to calculate rating similarity. We can compute the first similarity measure between user a and b as follows:

$$PR_{ab} = \frac{\sum_{e \in I(a,b)} (r_{ae} - \overline{r_a})(r_{be} - \overline{r_b})}{\sqrt{\sum_{e \in I(a,b)} (r_{ae} - \overline{r_a})^2} \sqrt{\sum_{e \in I(a,b)} (r_{be} - \overline{r_b})^2}} \tag{1}$$

$$Sim_{ab}^{R} = e^{-\omega PR_{ab}} (\omega > 0) \tag{2}$$

Where ω is a predefined parameter. Given two users a and b, r_{ae} represents the user's rating of the item e and $\overline{r_a}$ represents the average score of user a. If they have commonly rated items, $I(a, b)$ represents the set of common items. $|I(a, b)|$ represents the size of the set. Here, we adopt an exponential function to transform users' rating difference into a similarity value. The greater the value of Sim, the greater the similarity.

For items, our formula has the same principle and form. Given two items c and d, we can compute the similarity measure between item c and d as follows:

$$PR_{cd} = \frac{\sum_{e \in I(c,d)} (r_{ec} - \overline{r_c})(r_{ed} - \overline{r_d})}{\sqrt{\sum_{e \in I(c,d)} (r_{ec} - \overline{r_c})^2} \sqrt{\sum_{e \in I(c,d)} (r_{ed} - \overline{r_d})^2}} \tag{3}$$

$$Sim_{cd}^{R} = e^{-\omega PR_{cd}} (\omega > 0) \tag{4}$$

Feature Similarity Measure
In e-commerce websites, users and merchandise have many characteristics besides the shopping records. For example, mobile phones have many features such as brand, price, color and screen size, etc. These features are just as important as the rating matrix but often ignored by similarity measure. We decided to use similarity search on graph to deal with these similarity measure [26]. First of all, we use user features and commodity features to create two graphs. Users, items and their features are the nodes on the graph. An edge indicates that the user/item owns the feature. If two nodes have similar neighbors

in the network, we think they are similar. We can compute the second similarity measure between node a and b as follows:

$$Sim_{ab}^{F} = \frac{\sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} Sim^{F}\left(I_i(a), I_j(b)\right)}{|I(a)||I(b)|} \tag{5}$$

$$Sim_{ab}^{F} = 0, ifI(a) = \emptyset \; or \; I(b) = \emptyset \tag{6}$$

$$Sim_{aa}^{F} = 1 \tag{7}$$

$$Sim_{ab}^{F} = Sim_{ba}^{F}, \text{ symmetric} \tag{8}$$

Where $I(a)$ represents the entry neighborhood of node a, $|I(a)|$ represents the size of the neighborhood. This recursive algorithm is the same on two graphs, so we can express it uniformly.

Time Similarity Measure

With the advent of the era of mass information that consists of great time span data, the importance of temporal information is highlighted. Intuitively, the older the data is, the less time weight will be in similarity calculation, so the conventional research always use the forgetting curve to model the time factor. However, the time difference between different events or behaviors also contains important information. Sequence can represent relative time difference, but absolute time difference is often ignored. In the case of movies, we can measure the similarity of two movies by the number of people watching them together. But there are two difficult problems in the actual calculation. First of all, for many similar films, we can no longer distinguish the similarity differences in detail by traditional algorithms. Secondly, most of the time, what we are looking for is the next movie that we will see immediately after this movie, rather than the movie that will be seen eventually. So we created a model using time difference to compute similarity. The basic idea is that the closer the two movies are viewed, the more relevant they are. Take another example of a computer journal, the time interval between the author's papers published on TOC and TPDS is very short. Therefore, we can think that TOC and TPDS themes are very similar, and the level is very close. In order to solve the above problems, we created a time similarity measure model as follows:

$$\Delta T_{c,d|u_i} = \frac{\sum_{j=1}^{\mathrm{k}_{cd}} g\left(t_{c|u_i} - t_{d|u_i}\right)}{\mathrm{k}_{cd} t_m} \tag{9}$$

$$\Delta T_{c,d} = \frac{|I(c)||I(d)|}{|I(c,d)|} \cdot \frac{\sum_{i=1}^{|I(c,d)|} \Delta T_{c,d|u_i}}{|I(c,d)|} \tag{10}$$

$$Sim_{cd}^{T} = e^{-\mu \Delta T_{c,d}} (\mu > 0) \tag{11}$$

Where μ is a predefined parameter. Take shopping behavior as an example, $\Delta T_{c,d|u_i}$ represents the average time interval between the purchase of items c and d by user u_i. $t_{c|u_i}$ represents the time when user u_i purchases commodity c at one time. t_m represents the average time interval of user u_i's shopping behavior. Because users may buy the same product multiple times, k_{cd} represents the number of times user u_i purchases commodity c. $g\left(t_{c|u_i} - t_{d|u_i}\right)$ indicates that for each $t_{c|u_i}$, we select the closest $t_{d|u_i}$ to calculate the time difference. When there are two closest times, select the one that can get a positive value. $I(c, d)$ represents a set of users who jointly purchase two items. $|I(c, d)|$ represents the size of the set. $I(c)$ represents a set of users who purchased item c. $\Delta T_{c,d}$ represents the average time interval between the purchase of items c and d by all common users. The first half of the formula ensures that hot users/items do not affect the accuracy of the formula in Eq. 10. Here, we adopt an exponential function to transform users' rating difference into a similarity value.

In the later experiments, we verified the superiority of our innovative similarity algorithm. Finally, we combine the three categories of similarity:

$$Sim = \alpha Sim^R + \beta Sim^F + \gamma Sim^T \tag{12}$$

Where α, β and γ are the weights to control the importance of the three parts and Sim is the similarity we finally get.

3.2 Sequence Completion with Similarity Measure

First of all, given a cold start user a, we design a formula to calculate the heterogeneous similarity between user a and item c, so as to get the most similar top-N items with user a, and restrict these items from the target domain.

$$Sim_{ac} = \frac{\sum_{i \in I_c} Sim_{ai}}{|I(c)|} \tag{13}$$

Then, we change the Sim^T model slightly, and change the absolute time to the relative time, we can get a similarity model Sim^{RT} and ΔT^{RT} about the relative time.

Given a rating sequence of user a, j is an element in the sequence and i is an element in the set of top-N. The filling fraction formula is as follows:

$$f_{ij} = \frac{1}{n_i} Sim_{ai} Sim_{ji}^{RT} \tag{14}$$

Find the largest f_{ij}, fill its corresponding element i into the sequence, and fill in the front or back of position j. The front and back are determined by the positive and negative of ΔT^{RT}. n_i indicates the number of times the element i has been filled in the sequence.

Finally, the sequence is updated and the filling fraction is calculated repeatedly. The algorithm is repeated l times and $l < N$. The rating scores are calculated together as follows:

$$r_{ai} = \overline{r_a} + \tau \sum_{k=1}^{m} Sim_{ak}^R (r_{ki} - \overline{r_k}) \tag{15}$$

Where τ is a predefined parameter, and $\tau > 0$. Finally, the filling sequence S' is obtained. The pseudocode of the proposed sequence completion algorithm is shown in Algorithm 1.

Algorithm 1 sequence completion

```
Input  : $S_{av} = (s_{av_i})$: user a rating sequence for items.
         $I(a)$: the set of most similar top-N items with user $a$.
         $l$: the sequence length that we set, $l < N$.
Output: $S'_{av}$: the filling sequence we obtained.
1:   while $k \le l$ do
2:     for each item $j$ in $S_{av}$ do
3:       for each item $i$ in $I(a)$ do
4:         $f_{ij} = \frac{1}{n_i} Sim_{ai} Sim_{ji}^{RT}$;
5:     $f_{cd} = max(f_{ij})$;
6:     if $\Delta T_{cd}^{RT} > 0$ then
7:       Fill element $c$ into the sequence, immediately after $d$;
8:     else Fill element $c$ into the sequence, ahead of $d$;
9:     $k++$;
10:  return $S'_{av}$;
```

3.3 Sequential Recommendation

Through Long Short-Term Memory network (LSTM) [28], we finally get the recommended results for cold start users. We choose Cross Entropy Loss to find the error.

4 SCLSTM with Improved LSTM

Another model is to improve LSTM by building the connection between the output layer and the input layer of the next time step. Complete the sequence by adding this input logic unit. We can see the architecture of the model in Fig. 4.

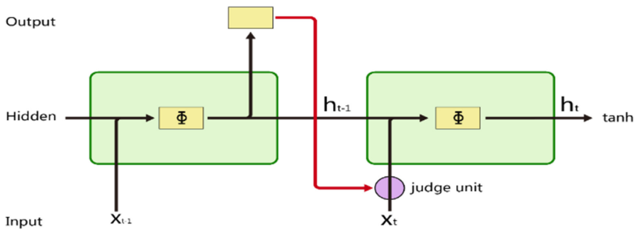

Fig. 4. Improved LSTM framework

By adding a judgment unit, we decide whether to take the output of this step as the input of the next step, so as to move the whole sequence backward. The algorithm uses the filling fraction formula in the previous chapter. The formula of judgment unit is as follows:

$$Ip_t = \begin{cases} Op_{t-1}, \; if \; W_i f_{Op_{t-1}x_{t-1}} + b_i \ge W_j f_{x_t x_{t-1}}, \; then \; t++; \\ x_t, \quad if \; W_i f_{Op_{t-1}x_{t-1}} + b_i < W_j f_{x_t x_{t-1}} \end{cases} \tag{16}$$

Where Ip_t represents the input of time step t, Op_{t-1} represents the output of time step $t-1$ and x_t represents the sequence element of time step t. f_{ij} is the filling fraction formula in

Eq. 14. $t++$ means to insert Op_{t-1} into the sequence, then the remaining elements move backward in turn. W_i and W_j is the weight parameter and b_i is the deviation parameter. They were trained together with other parameters of LSTM. In the process of propagation of neural networks, when $Ip_t = Op_{t-1}$, the propagation actually stops. Therefore, the number of iterations must be greater than the length of the maximum sequence.

5 Experiments

5.1 Experiments Settings

We use the Amazon dataset [29] to evaluate the performance of our model and baselines. There are ratings, simple attribute and reviews in total spanning from May 1996 to July 2014. It has 21 categories of items and we choose the three most widely used categories in cross-domain recommendation to perform the experiment. In order to gain better experiment performance, we filter the content in the dataset. We grab data from IMDb[2] and Google[3] to supplement features of users and items. The statistics of the two datasets are given in Table 1. We compare our model with the following baselines:

Table 1. The statistics of the two datasets

	Dataset 1		Dataset 2	
Domain	Movies	Books	Movies	CDs
Users	3479		4237	
Items	3983	2473	4302	5766
Ratings	19011	13002	43658	30263
Density	0.00193	0.00132	0.00176	0.00122

CMF: Collective Matrix Factorization (CMF) [27] tends to incorporate different sources of information by simultaneously factorizing multiple matrices.

EMCDR: This model [25] adopts matrix factorization to learn latent factors first and then utilize an MLP network to map the user latent factors.

Markov-RS: Markov chain-based sequential recommender systems [10] adopt Markov chain models to model the transitions over user-item interactions in a sequence, for the prediction of the next interaction.

LSTM: Given a sequence of historical user-item interactions, an RNN-based sequential recommender system [28] tries to predict the next possible interaction by modelling the sequential dependencies over the given interactions.

[2] https://www.imdb.com/.
[3] https://www.google.com/.

5.2 Evaluation Metric

In both data sets, we choose the domain with sparse data as the target domain. We select some users randomly in the target domain and hide their information as cold start users. In our experiments, we set the proportions of cold start users as 70%, 50% and 30% of the initial users respectively. The proportion is denoted as ϕ. We adopt Root Mean Square Error (RMSE) and Hit Ratio defined as follows as the evaluation metrics.

$$\text{RMSE} = \sqrt{\sum\nolimits_{r_{ac} \in I_{test}} \frac{\left(r_{ac} - \widehat{r_{ac}}\right)^2}{|I_{test}|}} \tag{17}$$

Where I_{test} is the set of test ratings. r_{ac} denotes an observed rating in I_{test}. $\widehat{r_{ac}}$ represents the predictive value of r_{ac}. $|I_{test}|$ is the number of test ratings.

$$\text{Hit Ratio} = \frac{\sum_u G(T_u \in R(u, t))}{|U|} \tag{18}$$

Where $G(\cdot)$ is an indicator function, $R(u, t)$ is a set of items recommended to user u at a specified time period t, T_u is the test item that user u accessed at a specified time period t and $|U|$ is size of all test sets. If the test item appears in the recommendation set, we call it a hit.

5.3 Experimental Results

The experimental results of RMSE on "Movies & Books" are shown in Table 2, and the results on "Movies & CDs" are presented in Table 3. The best performance of these models is shown in boldface. Because the data in the movie domain is relatively dense, we regard the movie domain as an auxiliary domain in both datasets. Our approaches are SCLSTM with similarity measure (SCLSTM1) and SCLSTM with improved LSTM (SCLSTM2). The parameters α, β and γ were finally determined to be 0.3, 0.1 and 0.6 in Eq. 14.

Table 2. Recommendation performance on "Movies & Books"

	RMSE		
ϕ	70%	50%	30%
CMF	1.4621	1.3305	1.2648
EMCDR	1.3583	1.0048	0.9496
Markov-RS	1.4365	1.4008	1.3701
LSTM	1.2543	1.1568	0.9970
SCLSTM1	**0.9477**	**0.9432**	0.9376
SCLSTM2	0.9951	0.9765	**0.9320**

Table 3. Recommendation performance on "Movies & CDs"

	RMSE		
ϕ	70%	50%	30%
CMF	1.6255	1.6118	1.6032
EMCDR	1.6753	1.1238	1.1494
Markov-RS	1.4213	1.4077	1.3653
LSTM	1.2775	1.2203	1.0988
SCLSTM1	**1.1380**	1.0776	1.0152
SCLSTM2	1.2203	**1.0377**	**0.9961**

We evaluate the performance of different models under different values of ϕ by RMSE model. From Tables 2 and 3, one can draw the conclusion that SCLSTM1 and SCLSTM2 are superior to all the state-of-the-art methods in cross-domain recommendation for cold start users. LSTM, SCLSTM1 and SCLSTM2 all perform better than CMF and EMCDR which proves the effectiveness of deep learning methods in cross-domain recommendation, even though the algorithm is not specially designed for cross-domain recommendation. With the increasing sparsity of data, the efficiency of all algorithms has declined. But we can see that the efficiency of SCLSTM1 fall less than those of other algorithms. Its performance is the most stable of all algorithms. When the data is denser, SCLSTM2 performs better than SCLSTM1. When the data becomes sparse, the performance of SCLSTM1 begins to exceed that of SCLSTM2. This is closely related to the advantage of SCLSTM1 as an algorithm based on similarity measure.

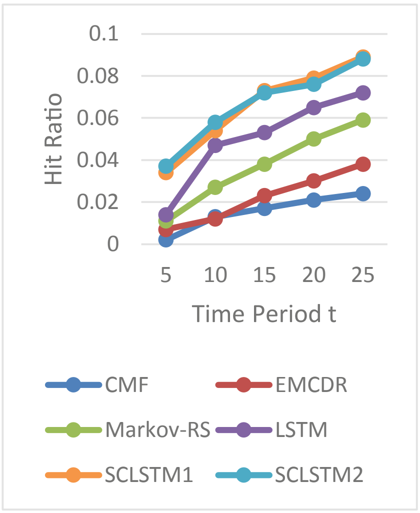

Fig. 5. Hit Ratio on "Movies & Books"

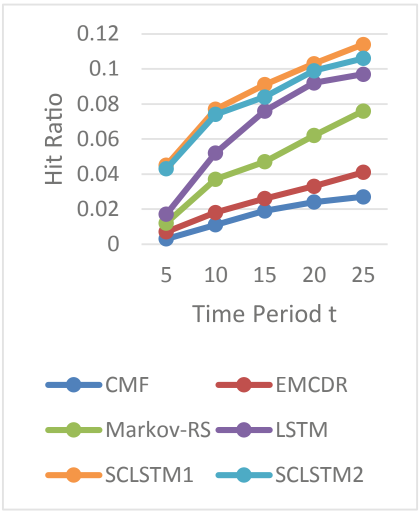

Fig. 6. Hit Ratio on "Movies & CDs"

Distinguished from other recommender systems, the most important feature of sequence recommendation is that its input and output are sequences. With this, we can not only recommend, but also recommend the right products at the right time. RMSE can't express this feature, but Hit Ratio can. Specified time period t represents the continuous t-times shopping behavior. We predict the t-times shopping information, and then compare it with the real t-length sequence. If there is one data coincidence, it means one hit. For the non-sequence recommendation algorithms CMF and EMCDR and the sequence recommendation algorithm Markov-RS which cannot output sequence data, we use Top-t data instead of prediction data. Figure 5 and 6 show that the results of Hit Ratio vary with t while $\phi = 30\%$. As you can see from the graph, the smaller the t, the greater the ratio of our two algorithms over other algorithms. With the increase of ϕ, t is fixed to 5 because of the sparse data in Fig. 7 and 8. Combining these four graphs, we can see that our two algorithms show great superiority in Hit Ratio.

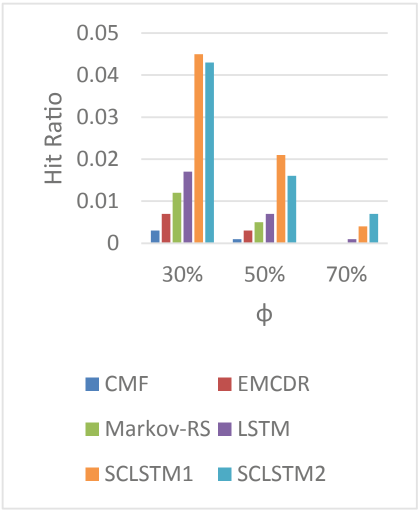

Fig. 7. Hit ratio on "Movies & Books" with t = 5

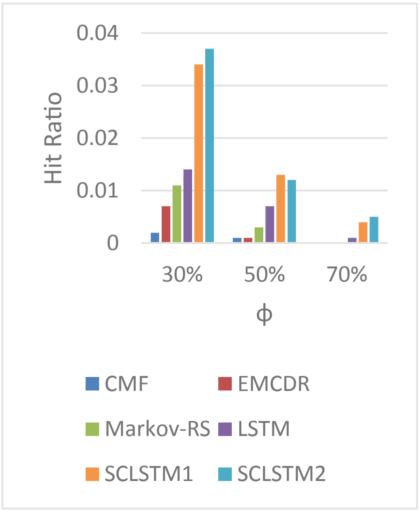

Fig. 8. Hit ratio on "Movies & CDs" with t = 5

Finally, we use collaborative filtering with different similarity algorithms to predict movie ratings in the Amazon movie dataset and utilize RMSE to evaluate the performance of all similarity algorithms. The similar algorithms we compare are Euclidean Distance (ED), Chebyshev Distance (CD), Cosine Similarity (CS), Personalized PageRank (PPR), SimRank (SR) [26], Jaccard Coefficient (JC) and our Time Similarity Measure (TS). The experimental results are shown in Fig. 9 and our algorithm shows great superiority.

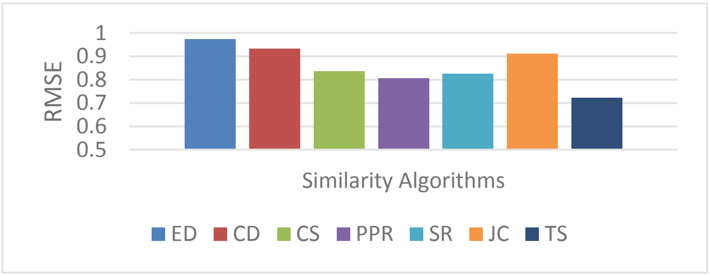

Fig. 9. Similarity algorithm performance

6 Related Work

Sequential Recommender Systems. Existing works about sequential recommender systems (SRSs) mostly consist of traditional sequence models [8–10], latent representation models [11–15], and deep neural network models [16–21].

Yap [8] proposed a sequential pattern-based recommender system which can mine frequent patterns on sequence data. Garcin [9] proposed a method of directly calculating the Markov chain transition probability based on the explicit observations. Feng [10] embedded the Markov chains into a Euclidean space and then calculates the transition probabilities between interactions based on their Euclidean distance.

Factorization machine-based SRSs usually utilize the matrix factorization or tensor factorization to factorize the observed user-item interactions into latent factors of users and items for recommendations [11, 12]. Such methods presents challenges in the face of data sparsity. Embedding-based SRSs learn a latent representations for each user and item for the subsequent recommendations by encoding all the user-item interactions in a sequence into a latent space. Specifically, some works take the learned latent representations as the input of a network to further calculate an interaction score between users and items, or successive users' actions [13, 14], while other works directly utilize them to calculate a metric like the Euclidean distance as the interaction score [15].

Deep neural networks nearly dominate SRSs in the past few years. Wu [16] proposed a method of capturing the long-term dependencies in a sequence with long short-term memory (LSTM) model while Hidasi [17] utilized gated recurrent unit (GRU) model and Quadrana [18] utilized hierarchical RNN. Both models are based on the improvement of recurrent neural network (RNN). A few works [19, 20] developed convolutional neural networks (CNN)-based SRSs. They first put all sequence data into a matrix, and then treat such a matrix as an image in the time and latent spaces. However, due to the limited sizes of matrix dimension CNN-based SRSs cannot effectively capture long-term dependencies. This is an obvious defect for sequence recommendation. Wu [21] transformed sequence data into directed graphs by mapping each sequence to a path and taking each interaction as a node in the graph, and then utilized graph neural network (GNN)-based SRSs to commendation.

Cross-Domain Recommender Systems. Cross-domain recommender systems have gained increasing attention in recent years. Existing studies [4–7] including the knowledge aggregation-based cross-domain recommender systems and the knowledge transfer-based cross-domain recommender systems and the latter methods is the focus of current research. Pan [22] proposed an adaptive models sharing potential features between two domains. Unlike adaptive algorithms, Pan [23] proposed a cooperative algorithms by learning potential features simultaneously between two domains, and optimizing a common objective function. Li [24] proposed a model based on rating patterns transfer.

7 Conclusion and Future Work

In this paper, we propose a Long Short-Term Memory with Sequence Completion model for cross-domain sequential recommendation. We first construct the sequence and supplement it in which two methods are proposed. Then we use LSTM to complete sequential recommendation. Experimental results on two real datasets extracted from Amazon transaction data demonstrate the superiority of our proposed models against other state-of-the-art methods. The current context of a user or commodity may greatly affect the user's choice of goods. When making recommendations, this should be taken into account. Therefore context-aware cross-domain sequential recommendations would be an important direction in our future works.

Acknowledgments. This work was supported by Shandong Provincial Key Research and Development Program (Major Scientific and Technological Innovation Project) (No. 2019JZZY010105), and NSF of Shandong, China (No. ZR2017MF065).

References

1. Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In: AAAI 2017, pp. 1309–1315 (2017)
2. Kang, W.-C., Wan, M., McAuley, J.J.: Recommendation through mixtures of heterogeneous item relationships. In: CIKM 2018, pp. 1143–1152 (2018)
3. Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q.Z., Orgun, M.A.: Sequential recommender systems: challenges, progress and prospects. In: IJCAI 2019, pp. 6332–6338 (2019)
4. Song, T., Peng, Z., Wang, S., Fu, W., Hong, X., Yu, P.S.: Review-based cross-domain recommendation through joint tensor factorization. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10177, pp. 525–540. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55753-3_33
5. Wang, X., Peng, Z., Wang, S., Yu, Philip S., Fu, W., Hong, X.: Cross-domain recommendation for cold-start users via neighborhood based feature mapping. In: Pei, J., Manolopoulos, Y., Sadiq, S., Li, J. (eds.) DASFAA 2018. LNCS, vol. 10827, pp. 158–165. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91452-7_11
6. Cantador, I., Fernández-Tobías, I., Berkovsky, S., Cremonesi, P.: Cross-domain recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 919–959. Springer, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_27

7. Ignacio, F.-T., Cantador, I., Kaminskas, M., Ricci, F.: Cross-domain recommender systems: a survey of the state of the art. In: Spanish Conference on Information Retrieval, vol. 24 (2012)
8. Yap, G.-E., Li, X.-L., Yu, P.S.: Effective next-items recommendation via personalized sequential pattern mining. In: Lee, S.-G., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012. LNCS, vol. 7239, pp. 48–64. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29035-0_4
9. Garcin, F., Dimitrakakis, C., Faltings, B.: Personalized news recommendation with context trees. In: RecSys 2013, pp. 105–112 (2013)
10. Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y.M., Yuan, Q.: Personalized ranking metric embedding for next new POI recommendation. In: IJCAI 2015, pp. 2069–2075 (2015)
11. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: WWW 2010, pp. 811–820 (2010)
12. Hidasi, B., Tikk, D.: General factorization framework for context-aware recommendations. Data Min. Knowl. Discov. **30**(2), 342–371 (2015). https://doi.org/10.1007/s10618-015-0417-y
13. Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., Cheng, X.: Learning hierarchical representation model for nextbasket recommendation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 403–412 (2015)
14. Wang, S., Hu, L., Cao, L., Huang, X., Lian, D., Liu, W.: Attention-based transactional context embedding for next-item recommendation. In: AAAI 2018, pp. 2532–2539 (2018)
15. He, R., Kang, W.-C., McAuley, J.J.: Translation-based recommendation: a scalable method for modeling sequential behavior. In: IJCAI 2018, pp. 5264–5268 (2018)
16. Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A.J.: How Jing: recurrent recommender networks. In: WSDM 2017, pp. 495–503 (2017)
17. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: ICLR (Poster) (2016)
18. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: RecSys 2017, pp. 130–137 (2017)
19. Tang, J., Wang, K.: Personalized top-N sequential recommendation via convolutional sequence embedding. In: WSDM 2018, pp. 565–573 (2018)
20. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: WSDM 2019, pp. 582–590 (2019)
21. 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, vol. 33, pp. 346–353 (2019)
22. Pan, W., Xiang, E.W., Liu, N.N., Yang, Q.: Transfer learning in collaborative filtering for sparsity reduction. In: AAAI 2010 (2010)
23. Pan, W., Liu, N.N., Xiang, E.W., Yang, Q.: Transfer learning to predict missing ratings via heterogeneous user feedbacks. In: IJCAI 2011, pp. 2318–2323 (2011)
24. Li, B., Yang, Q., Xue, X.: Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In: IJCAI 2009, pp. 2052–2057 (2009)
25. Man, T., Shen, H., Jin, X., Cheng, X.: Cross-domain recommendation: an embedding and mapping approach. In: IJCAI 2017, pp. 2464–2470 (2017)
26. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: KDD 2002, pp. 538–543 (2002)
27. Singh, A.P., Gordon, G.J.: Relational learning via collective matrix factorization. In: KDD 2008, pp. 650–658 (2008)
28. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

29. McAuley, J.J., Pandey, R., Leskovec, J.: Inferring networks of substitutable and complementary products. In: KDD 2015, pp. 785–794 (2015)
30. Wang, S., Hu, X., Yu, P.S., Li, Z.: MMRate: inferring multi-aspect diffusion networks with multi-pattern cascades. In: KDD 2014, pp. 1246–1255 (2014)