Recommendation Method by Fusing Interactive Sequences and Class Labels

Chuanchuan Zhao, Jinguo You and Jiaman Ding

June 15, 2019

Recommendation Method by Fusing Interactive Sequences and Class Labels*

Chuanchuan Zhao[1], Jinguo You[1,2](✉), and Jiaman Ding[1]

[1] Kunming University of Science and Technology, Yunnan Province 650504, China
[2] Computer Technology Application Key Lab of Yunnan Province, Yunnan Province 650504, China
jgyou@126.com

Abstract. The deep learning method introduces the recommendation system, which can solve the problem that the traditional recommendation system cannot capture the user's evolution preferences over time, but the existing sequence recommendation only considers the sequential similarity between the items and does not consider the content feature information between the items. This paper proposes a recommendation method by fusing interactive sequences and class labels based on the deep bidirectional LSTM model. The method is divided into two main parts: improved item embedding and deep bidirectional LSTM model preference learning. First, the item2vec model is used to embed the user interaction sequence into a low-dimensional space vector representation, and a category label vector is added for each embedded item vector. Secondly, the embedded item vector is input into the bidirectional LSTM model to learn the user's preference vector. Finally, the recommendation list is generated by the preference vector to calculate the most similar items in the embedded space. By experimenting on the real data set and comparing with the advanced methods, we proves that the method has improved the evaluation index recall rate and the Mean Reciprocal Rank compared with the comparison method, which better solves the problem of user interest evolution over time.

Keywords: Interactive Sequences, Class Label, Deep Learning, Deep Bidirectional LSTM.

1 Introduction

In general, the recommendation list is generated based on the user's own attribute preferences, the basic attribute characteristics of the item, and other auxiliary information. The models of the recommendation system are roughly divided into three types: content-based recommendation, collaborative filtering, and hybrid recommendation [1]. However, in the practical application of real

* This work is supported by the Natural Science Foundation of China (61462050, 61562054), the Natural Science Foundation of Yunnan Province (KKSY201603016).

life, the above-recommended algorithms have some serious problems: cold start problem and data sparseness problem.

Recent years have seen a surge in approaches that combines deep learning into recommendation systems [2]. Deep learning not only well trains some non-linear or non-trivial implicit relationships between users and items, but also efficiently encodes the underlying seemingly complex abstract meanings of the network into higher-level data representations. In addition, it has a more important performance: deep learning can learn the intricate relationship of data itself from the rich data resources that are easily available. Due to these high-performance features of deep learning, its application in the recommended system field has gradually become more widespread in recent years, but there are also corresponding problems. The sequence recommendation method in [3] only adopts the sequential similarity between items, and does not consider the correlation between items. A good recommendation system should consider the content information between items. In order to solve this problem, the content of this item will be embedding information into the item vector together, the embedding not only better learns the user's preferences, but is more accurate when calculating project similarities.

To address this problem, this paper proposes a Recommendation Method by Fusing the Interactive Sequences and Class Labels (FISCL). The method introduces the user's preference and deep recurrent neural network, which input the sequence that consists of the user interacted items into the recurrent neural network. After multiple training, the last layer of the method outputs the user's preference vector. Finally, all the item is ranked by the similarity between the preference vector and the embedded vectors, and the Top-k items are selected as the recommendation list of the user.

2 Recommendation Method by Fusing Interactive Sequences and Class Labels

The deep recurrent neural network performs well in the NLP. In the natural language processing, a deep threshold recurrent network is used. Since the LSTM in the recurrent neural network is a model that performs well in sequence prediction problems, this paper employs the deep two-way LSTM to learn user preference information in a deeper level to learn user preferences.

Our recommendation method includes two phases: item embedding and user preference learning.

2.1 Item Embedding

Given all user interaction item sets $S = (S_1, S_2, S_3, \ldots, S_m)$, where S_i represents user i's interaction sequence $S_i = \{I_1, I_2, I_3, \ldots, I_n\}$, the purpose of item embedding is to generate low-dimensional item vectors for each item. This paper only selects sequences that show feedback on user preferences. For example, if a user scores a low-interest item, the user is not interested in the item. Traditional item embedding often only considers the second-order correlation between items and does not consider the relationship between item attributes and content. This paper embeds the label category of the item into the item vector, which can better calculate the relevance of the item and learn the user's preferences. This paper uses the item2vec [4] deep learning model to generate embedded vectors for the project.

Item2vec is one of the important extensions of Skip-gram and negative sampling [5] for item embedding for item-based collaborative filtering recommendations. In order to introduce the class label into the method, this paper has made some improvements to item2vec. Similar to word2vec, this paper treats each item as a word. The sequence of user-interacted items is treated as a sentence, and each item is embedding into a vector of fixed dimensions. Each user has its own interacted sequence and is different from each other. Each user has its own sequence of interactions and is different from each other. Finally, by embedding each user's sequence of interactions to obtain a fixed-dimensional interaction vector, the closer the vectors are, the closer they are in the embedded space.

Given a user's sequence of interactions, the Skip-gram goal is to maximize the following objective functions:

$$\frac{1}{M}\sum_{i=1}^{M}\sum_{j\neq i}^{M}\log p\left(I_j|I_i\right) \tag{1}$$

Where M is the length of the sequence of interactions, $p\left(I_j|I_i\right)$ is the SoftMax function:

$$p\left(I_j|I_i\right)=\sigma\left(I_i^T I_j\right)\prod_{k}^{N}\sigma\left(-I_i^T I_k\right) \tag{2}$$

Where $\sigma(x)$ is a commonly used Sigmoid activation function and N is the number of negative samples in the positive sample. By item embedding, we got $S_i = \{V_1, V_2, V_3, \ldots, V_n\}$.

2.2 User Preference Learning

Inspired by the literature [6, 7], the user's sequence of interactions is treated as a sentence, and each item can be thought of as a word, using deep recurrent neural network to learn the relevance of each item in the sequence of interactions to the adjacent item. This paper improves LSTM to deep bidirectional LSTM (ie DBi-LSTM), as shown in Figure 1, enabling the model to better utilize forward and backward contexts representation and the deep recurrent neural network can be better. And deep recurrent neural networks also can better abstract the characteristics of the user.

Introduction to the DBi-LSTM model: RNN is able to play a good role in predicting the next target of the sequence. Based on this, this paper proposes a DBi-LSTM model structure for a recommendation.

Figure 1 shows a model of preference modeling, which has a double hidden layer, and information of each upper layer in the structure is provided by its lower layer. As the picture shows, in the network structure, the previous time step generates a set of parameters and passes the set of parameters to the interneurons in the same Bi-LSTM layer at a later time step t. At the same time, the interneuron needs to receive two sets of related parameters from the previous layer of the Bi-LSTM hidden layer in the time step t; the input sequence of each hidden layer in the model starts from two directions: from left to right, from right to left.

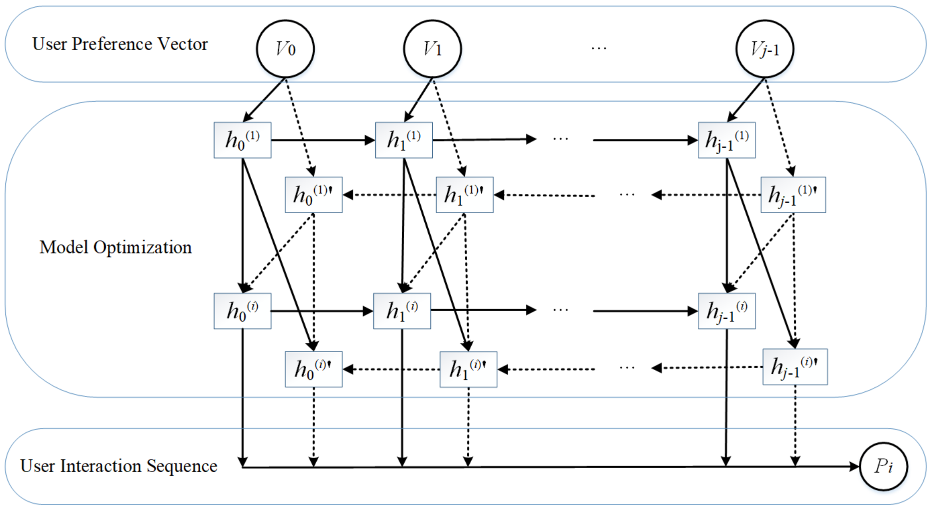

Fig. 1. Preference Learning Model.

Fig. 1. The relational formula of the DBi-LSTM structure as in Equations 3, 4. At the same time step t, each output of the layer Bi-LSTM of $i-1$-th layer serves as an input to each intermediate neuron of the i-th layer. At each time step in the training model, the result produced by the hidden layer propagation process by connecting all the input parameters is used as the output P of the last hidden layer (Equation 5).

$$\overrightarrow{\boldsymbol{h}}_t^{(i)} = f\left(\overrightarrow{\boldsymbol{W}}^{(i)}\overrightarrow{\boldsymbol{h}}_t^{(t-1)} + \overrightarrow{\boldsymbol{V}}^{(i)}\overrightarrow{\boldsymbol{h}}_{t-1}^{(i)} + \overrightarrow{\boldsymbol{b}}^{(i)}\right) \tag{3}$$

$$\overleftarrow{\boldsymbol{h}}_t^{(i)} = f\left(\overleftarrow{\boldsymbol{W}}^{(i)}\overleftarrow{\boldsymbol{h}}_t^{(i-1)} + \overleftarrow{\boldsymbol{V}}^{(i)}\overleftarrow{\boldsymbol{h}}_{t+1}^{(i)} + \overleftarrow{\boldsymbol{b}}^{(i)}\right) \tag{4}$$

$$\boldsymbol{P} = concat\left(\overrightarrow{\boldsymbol{h}}_t^{(i)}, \overleftarrow{\boldsymbol{h}}_t^{(i)}\right) \tag{5}$$

Where $\overrightarrow{\boldsymbol{W}}^{(i)}$, $\overrightarrow{\boldsymbol{V}}^{(i)}$ and $\overrightarrow{\boldsymbol{b}}^{(i)}$ are weight matrix and offset vector generated in forward propagation at the i-th layer of the model; $\overleftarrow{\boldsymbol{W}}^{(i)}$,$\overleftarrow{\boldsymbol{V}}^{(i)}$ and $\overleftarrow{\boldsymbol{b}}^{(i)}$ are weight matrix and offset vector generated in backward propagation at the i-th layer of the model; $\boldsymbol{P}$ is output vector; $\overset{\rightarrow(i)}{\boldsymbol{h}_t}$ and $\overset{\leftarrow(i)}{\boldsymbol{h}_t}$ are respectively the intermediate representation of the past and the future is used to discriminate the input vector.

The i-th layer propagation of the model is based on the hidden state $\overset{\rightarrow(i-1)}{\boldsymbol{h}_t}$ of the current moment of the previous layer and the hidden state $\overset{\rightarrow(i)}{\boldsymbol{h}_{t-1}}$ of the previous moment of the current layer to calculate the forward direction of the hidden state $\overset{\rightarrow(i)}{\boldsymbol{h}_t}$ of the current layer at the current moment; In contrast, backward propagation requires the hidden state $\overset{\leftarrow(i-1)}{\boldsymbol{h}_t}$ of the current layer and the future state $\overset{\leftarrow(i)}{\boldsymbol{h}_{t+1}}$ of the current layer to calculate the hidden state $\overset{\leftarrow(i)}{\boldsymbol{h}_t}$ of the update reverse. Thus, each hidden representation can be computed using a tandem calculation function $concat\left(\overset{\rightarrow(i)}{\boldsymbol{h}_t}, \overset{\leftarrow(i)}{\boldsymbol{h}_t}\right)$ that concatenates the forward and backward hidden representations. The last layer of the hidden layer outputs the preference vector through the fully connected layer.

The embedded vector of each item (the vector after the user's watching interacted sequence is embedded) is used as the input of the model. In the model training process, the mean square error (MSE) and Adagrad are used to learn the optimization model, so that the model can well learn the preferences of each user and better understand and represent the user's long-term stable preference.

3 Algorithms

In the entire process of our recommendation system, Alg. 1 is first executed and then Alg. 2 is performed, which is shown as follows.

Algorithm 1 Item Embedding

```
Input:
    All user interaction item sets $\boldsymbol{S} = (S_1, S_2, S_3, \ldots, S_m)$;
    The sequence of items $\boldsymbol{S_i} = \{I_1, I_2, \ldots, I_n\}$ that the user $\mathbf{U}_i$ has interacted with;
    Class label vector for each item, $\boldsymbol{L}$.
Output:
    A vector representation of per items in a low-dimensional space, $\boldsymbol{V}_i$.
1: Extract the item set $\mathbf{I}$ and the label set $\mathbf{L}$ corresponding to the item;
2: Extract the user's sequence of interactions from the user-item score and perform
   one-hot encoding on the item class label $\boldsymbol{L}$;
3: Use item2vec to learn the sequence of user interaction items to learn the embedded
   vector representation $\boldsymbol{V}_i$ corresponding to each item in the low-dimensional space.
4: return $\boldsymbol{V}_i$;
```

Alg. 2 is to learn the user preference and and finally generate a recommendation list.

Algorithm 2 User Preference Learning

```
Input:
    The first $j-1$ item set $S_i = \{V_1, V_2, V_3, \dots, V_n\}$ of all item sequences that the target user has interacted with.
Output:
    The preference vector $\boldsymbol{P}_i$ of the target user $\mathbf{U}_i$;
    A recommendation list.
1: Use Deep Bi-LSTM to learn user preferences vector $\boldsymbol{P}_i$ (adding multiple hidden layers to enhance the method's expressive ability);
2: Calculate the similarity between the preference vector $\boldsymbol{P}_i$ of the target user $\mathbf{U}_i$ and the low-dimensional vector $\boldsymbol{V}_j$ of each item learned in the low-dimensional space, and obtain the similarity $sim_{U_i,I_j}$ of the target user preference and the item: $sim_{U_i,I_j} = \boldsymbol{P}_i \cdot \boldsymbol{V}_j,\ i = 1, 2, \cdots, m; j = 1, 2, \cdots, n$
3: Filter out the item set $\boldsymbol{V}'$ that the target user does not interact with, sort according to the similarity $sim_{U_i,I_j}$ $(I_j \in I')$, recommend Top-k items to the target user, and generate a recommendation list.
4: return $\boldsymbol{P}_i$, list
```

4 Experimental Evaluation

This section first briefly introduces the real data set used, then introduces the evaluation criteria and comparison methods used in this method. Finally, the experimental results show that the experimental analysis and evaluation results are compared with other methods. The programming language used in this article is python.

4.1 Dataset

The dataset used in this method is the MovieLens [8] 10m dataset, which is a relatively classic dataset in the field of recommendation systems. It contains 710,054 ratings and 20 class label information for 71,567 users for 10,681 movies. The users selected in the data set are all users who have scored at least 20 movies. The score data set contains the timestamp of the user interaction.

The data set is divided into two parts: the training set and the test set. 90% of the data is randomly used as the training set to complete the training of the whole algorithm, and the remaining 10% of the data is used as the test set to measure the actual performance of the method. When training the method, the items with less than 3 training users in the training set are filtered from the interactive sequences, which makes the method performs better and more appropriate to the user's preference.

4.2 Benchmark Algorithms

In order to verify the validity of the proposed method, the following three algorithms are given as comparisons:

Item-based k-NN [9]: An item-based nearest neighbor recommendation. It is a collaborative filtering method that determines the target user's rating of the item, finds other items that are similar to the item , and infers his rating of the item based on the target user's rating of a similar item. In [9], the non-personalized variants of the item-based k-NN achieved good recommendation results in the sequence prediction task.

Exp. Dec. Item-based k-NN: An exponential decay component is added to the Item-based k-NN to punish items that were consumed early.

Matrix Factorization (MF): A typical model-based collaborative filtering recommendation algorithm that uses the rating information between users and items to predict the target user's rating of the item. However, MF only uses scoring data. Due to the scoring matrix is actually a very sparse matrix, MF has serious data sparsity problems, and it is a shallow model that cannot learn further features between users and items.

Seq. Matrix Factorization: is a method of adding a sequence based on MF and expanding to a sequence MF method.

Standard GRU [10]: Some recommended methods take advantage of information about user interaction behavior, for example, the literature [10] uses the standard GRU to embed the user interaction item and its corresponding actions into a model to learn the interaction mode of the user, and finally outputs the next possible interaction action of the user.

4.3 Evaluation Criterion

Two ranking-based evaluation indicators are used to evaluate the quality of the next recommendation:

Recall @k: It is the main evaluation indicator. Defined as the next item of the target user's actual interaction sequence appears in the Top-k recommendation list, which is the proportion of cases with the required items in the Top-k items in all test cases [11].

MRR @k: Another evaluation criterion is the Mean Reciprocal Rank, the average of the reciprocal ranking of the next item of the target user's actual interaction sequence in the recommendation list [12]. If it is higher k, the level is set zero.

$$MRR@k = \frac{1}{|Q|}\sum_{x=1}^{|Q|}\frac{1}{rank_x} \tag{6}$$

Where $|Q|$ is the number of items that the test centralized user actually interacts, and $rank_x$ is the item where the user actually interacts in the x-th position of the recommendation list.

In short, the higher the value of these two evaluation indicators, the better the recommended results of this method. (In this paper, we set k=20)

4.4 Experimental Results

The parameters of the three comparison algorithms are configured as follows: Item-based k-NN nearest neighbors is set to 100; Exp. Dec. Item-based k-NN has a decay constant set to 1; The potential factor of Matrix Factorization (MF) The number and number of iterations are set to 20 and 30, respectively; The window size of Seq. Matrix Factorization is set to 2; The Batch and iteration times of Standard GRU are set to 1000 and 10, respectively.

The experimental results on the real data set prove that the proposed method has certain practical significance. This paper compares the experimental results with other advanced methods on the same data set MovieLens, as shown in Table 1, 2, 3:

method Recommendation Performance Table 1 shows the comparison of all the comparison methods with the method on the two evaluation indicators Recall@20 and MRR@20.

Table 1. Performance comparison between this method and advanced method on MovieLens dataset.

	Recall@20	MRR@20
Item-based k-NN	0.12142	0.03639
Exp. Dec. Item-based k-NN	0.12853	0.04231
Matrix Factorization (MF)	0.07744	0.01192
Seq. Matrix Factorization	0.10730	0.01550
Standard GRU	0.15773	0.04730
FISCL	0.22634(+43.50%)	0.05634(+19.11%)

The proposed method produces better results than other comparison methods. Especially on the Recall@20 indicator, after adding the item content information, the recommended performance is significantly improved compared to the Standard GRU, which can be increased by up to 43.50%, and on the MRR@20 indicator, the performance increased by up to 19.11%.

The performance of the recommended system has been improved by adding time and sequence factors from the traditional method. It can be seen from the Item-based k-NN and Exp. Dec. Item-based k-NN that the calculation steps are the same, and the time factor can improve the performance. Prove that using sequence recommendations can better mine users' interest preferences. From the comparison of Matrix Factorization (MF) and Seq. Matrix Factorization, it is found that by adding user interaction sequence factors, user preferences can be better modeled, and the hidden correlation between items can be obtained from the sequence.

The proposed method is significantly improved compared with the standard sequence recommendation GRU. The key point is that the item embedding process combines the content information of the item and the design of the neural network, which not only considers the order similarity of the item, but also considers the item's content similarity, a deeper neural network can better learn user preferences.

Impact of Embedded Dimensions on method Performance In the method, the dimensions of the items embedded representation also affect the method's performance.

Table 2. The Impact of Embedded Dimensions on the Performance of This method.

Dimension	Recall@20	MRR@20
8	0.13587	0.02849
16	0.19895	0.04214
20	0.22634	0.05634
32	0.19895	0.04214

Table 2 shows the effect of the size of the embedded dimension on the performance of the proposed method in the item embedding process. From the experimental results in the table, it can be seen that when the embedding dimension is 8 dimensions, the embedded representation of the item does not reach the best, and the recommended performance of the trained method is relatively low, and the main evaluation index Recall@20 is 13.77% worse than the 16-dimensional result. When the embedding dimensions are 16 and 32, the two index values corresponding to the two dimensions are equal, and the performance of the method is improved compared to the 8-dimensional index value, but this does not mean that the higher the dimension, the recommended performance of the method is better. When the dimension is 20, the performance of the method reaches the highest value relative to other dimensions on the two indicators, and the recommended performance is relatively good.

Performance of Deep Neural Networks In order to prove the validity of the proposed deep neural network, the experimental results of single-layer bidirectional LSTM (Bi-LSTM) and deep bidirectional LSTM (DBi-LSTM) are compared experimentally, as shown in Table 3. It can be seen from the table that the deep model proposed is much improved compared to the single-layer model. From the two evaluation indicators Recall@20 and MRR@20, the recommended performance results for DBi-LSTM are 1.59% and 2.91% higher than Bi-LSTM, respectively. And the key is that deep neural networks can learn a deeper representation of preferences between user interaction sequences.

Table 3. Effect of Bidirectional LSTM with Different Layers on Evaluation Indicators.

	Bi-LSTM	DBi-LSTM
Recall@20	0.22280	0.22634(+1.59%)
MRR@20	0.05475	0.05634(+2.91%)

Our method is much better than other traditional advanced algorithms. The key lies in the application of deep learning. Deep learning can better mine the potential characteristics of users, deeply model the correlation between items, and improve the performance of recommender systems.

5 Related Work

Irsoy et al. [13] have shown through experiments that at each time , a bidirectional deep neural network with two hidden layers is used for the extraction evaluation and the efficiency of the evaluation expression is proved by experiments, wherein a hidden layer is used for positive Propagation (from left to right) and another layer for back-propagation (from right to left). In order to be able to maintain these two hidden layers well at all times, the network's weight and offset parameters consume twice as much memory space. Finally, the final classification result is generated by combining the result scores produced by the two-layer RNN hidden layer. Equations 7, 8 are used to calculate the hidden layer representation of the bidirectional RNN. The only difference between the two hidden layers is that they are recursively different through the corpus. Equation 9 is a comprehensive representation of the hidden learning in both directions, thereby predicting the possible classification relationship of the next word.

$$\overrightarrow{\boldsymbol{h}}_t = f\left(\overrightarrow{\boldsymbol{W}}_{x_t} + \overrightarrow{\boldsymbol{V}}\overrightarrow{\boldsymbol{h}}_{t-1} + \overrightarrow{\boldsymbol{b}}\right) \tag{7}$$

$$\overleftarrow{\boldsymbol{h}}_t = f\left(\overleftarrow{\boldsymbol{W}}_{x_t} + \overleftarrow{\boldsymbol{V}}\overleftarrow{\boldsymbol{h}}_{t-1} + \overleftarrow{\boldsymbol{b}}\right) \tag{8}$$

$$\hat{y}_t = g\left(\boldsymbol{U}\boldsymbol{h}_t + c\right) = g\left(\boldsymbol{U}\left[\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t\right] + c\right) \tag{9}$$

Where $\overrightarrow{\boldsymbol{W}}$, $\overrightarrow{\boldsymbol{V}}$ and $\overrightarrow{\boldsymbol{b}}$ are weight matrix and offset vector generated in forward propagation; $\overleftarrow{\boldsymbol{W}}$, $\overleftarrow{\boldsymbol{V}}$ and $\overleftarrow{\boldsymbol{b}}$ are weight matrix and offset vector generated in backward propagation; U is output matrix; $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are respectively the intermediate representation of the past and the future is used to discriminate the input vector, and c is the output deviation.

In [14], Feng Yong et al. proposed an MN-HDRM recommendation model. It first learned the user's interest changes in a short period of time through RNN and used FNN to learn the target users for a long period of time. Stable interest, and finally the fusion of two networks to construct a hybrid dynamic recommendation model of long-short interest multi-neural network, and achieved relatively superior performance.

Literature [15] proposed a convolutional LSTM network for predicting precipitation weather. The authors regard the precipitation prediction in the most recent period as a space-time sequence prediction problem and form a coding pre-diction by stacking multiple layers of convolutional LSTM layers. The structure of the end-to-end precipitation prediction training model is constructed.

H. Gao et al. [16] conducted deep research on the time-cycle mode that users sign in in terms of time non-uniformity and time continuity. Q Yuan et al. [17] incorporated time-cycle information into a user-based collaborative filtering frame-work for time-aware POI recommendations.

Although the above method uses deep learning to make recommendations, the content information of the item is not fully utilized. Based on this, this paper proposes a method for fusing interactive sequences and class labels based on the deep bidirectional LSTM model, which not only utilizes the user's sequence of interactions but also makes full use of the item's content information to explore deeper relationships between items.

6 Conclusion and Future Work

In order to improve the performance of the recommendation system, this paper proposes the recommendation method for fusing interactive sequences and class labels. Firstly, the user interaction sequence and the items class labels are embedded to obtain the user's sequence of interactions vector. Then, the sequence vector is input to the deep Bi-LSTM, the method is trained to obtain the preference vector of each user. Finally, the Top-k recommendation is made by calculating the similarity between the preference vector and the item vector. The experiment proves that the proposed method has better performance. In future work, we intend on adding more item content information for embedding, further representing the item, and combining the user's basic information to more accurately learn the user's preferences.

References

1. Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering 17, 6 (2005), 734749.
2. Zhang S , Yao L , Sun A , et al. Deep Learning based Recommender System: A Survey and New Perspectives[J]. ACM Computing Surveys, 2017.
3. Y. K. Tan, X. Xu, and Y. Liu. 2016. Improved Recurrent Neural Networks for Session-Based Recommendations. In DLRS 16. ACM, 1722.
4. Oren Barkan and Noam Koenigstein. 2016. Item2vec: Neural Item Embedding for Collaborative Filtering. In Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on. IEEE, 16.
5. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In Advances in Neural Information Processing Systems. 31113119.
6. Kiperwasser E , Goldberg Y . Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations[J]. 2016.
7. He M, Morimoto Y. Capturing Temporal Dynamics of Implicit Feedbacks for Collaborative Filtering by Using Deep Recurrent Neural Networks[J]. Bulletin of Networking, Computing, Systems, and Software, 2018, 7(1): 33-37.
8. Grouplens Homepage, https://grouplens.org/datasets/. Last accessed 21 May 2019
9. H. Soh, S. Sanner, M. White, and G. Jamieson. 2017. Deep Sequential Recommendation for Personalized Adaptive User Interfaces. In IUI 17. ACM, 589593.

10. B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2015. Session-Based Recommendations with Recurrent Neural Networks. In ICLR 16.
11. Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM 17). 14191428.
12. E. M. Voorhees. 1999. The TREC-8 question answering track report. In TREC 99. 7782.
13. Irsoy O, and Claire C. Opinion Mining with Deep Recurrent Neural Networks[C]//Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). 2014.
14. Feng Yong, Zhang Bei, Qiang Baohua, et al. MN-HDRM: A Novel Hybrid Dynamic Recommendation Model Based on Long-and-Short Interest Multiple Neural Networks[J]. Chinese Journal of Computers, 2019(1):16-28.
15. Shi X, Chen Z, Wang H, et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting[J]. 2015.
16. H. Gao, J. Tang, X. Hu, and H. Liu. Exploring temporal effects for location recommendation on location-based social networks. In RecSys, pages 93100, 2013.
17. Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In SIGIR, pages 363372, 2013.