Sequence-Based Recommendation with Convolutional Bidirectional LSTM Network

Hailin Fu, Jianguo Li(✉), Jiemin Chen, Yong Tang, and Jia Zhu

School of Computer Science, South China Normal University,
GuangZhou, China, 510000
{hailin,jianguoli,chenjiemin,ytang,jzhu}@m.scnu.edu.cn

Abstract. In modern recommendation systems, most methods often neglect the sequential relationship between items. So we propose a methodology named Convolutional Bidirectional Long Short-Term Memory (CBLSTM) network to capture the sequential feature of items. Then we use this methodology to build a sequence-based recommendation to predict what a user will choose next. By collecting consumed items of a user in a sequence with time ascending order, fitting the model with the last item as the label, the rest items as the features, we regard this recommendation assignment as a super multiple classification task. Once trained well, the output layer of our model will export the probabilities of the next items with given sequence. In the experiments, we compare our approach with several commonly used recommendation methods on a real-world dataset. Experimental results indicate that our sequence-based recommender can perform well for short-term interest prediction on a sparse, large, imbalanced dataset.

Keywords: Recommendation systems · Social media data mining · bidirectional recurrent neural network · convolutional neural network · deep learning.

1 Introduction

The primary assignments of recommendation systems faced with consist of two parts: ratings predicting and products recommendation. So to predict what a user will choose next given his consumed history is one of the crucial mission [15] in recommendation area. In many websites and applications, such as online electronic business, news/videos website, music/radio station, they need an excellent service for users to recommend what they will like in future. Existing recommenders mainly concentrate on finding the neighbor sets for users or items, or leveraging other explicit/implicit information (such as tags, reviews, item contents and user profiles) for neighborhood-aware. However, to the best of our knowledge, few works use the sequential feature of data to build recommender. We find that the sequence of data implicates much exciting and relevant information, for example in a video website, the user who watched "Winter is coming" (S01, E02 of Game of Thrones) will be more likely to watch "The

Kingsroad" (S01, E02 of Game of Thrones). Even at the 2011 Recsys conference, Pandora[1]'s researchers gave a speech about music recommendation and said they find many users consumed music in sequences.

Our work was inspired by the previous study of Siwei Lai et al.[13], where a neural network is proposed to capture the sequence of words in a sentence. We took a similar approach by considering one item as a word, the catalog of items as a vocabulary, and the historical consumed items of one user as a sentence, to capture the sequence of the user consumed items. The main contributions of our work are as follows:

- This paper firstly introduces Convolutional Bidirectional Long Short-Term Memory(CBLSTM) network to the domain of Recommendation System.
- We propose a novel Sequence-based recommendation framework with deep neural network, which can capture the sequential features of data, as well scales linearly with the number of objectives (both of users and items).

2 Related Work

2.1 Traditional methods in recommendation

There are three main classes of traditional recommendation system. Those are collaborative filtering systems, content-based filtering systems and hybrid recommendation systems [12]. Collaborative filtering [1–4] systems generate recommendations based on crowd-sourced input. They recommend for users by finding similar user groups, analyzing those similar user ratings for certain item, to generate preference prediction for this user in special products. Collaborative filtering algorithm can be generally classified into Memory-based [1, 2] collaborative filtering and Model-based [3–5]collaborative filtering. Memory-based collaborative filtering include User-based collaborative filtering [1] which evaluate the similarity between users by different users ratings on the item, making recommendations based on the similarity between users and Item-based collaborative filtering [2] which evaluate the similarity between items by the users rating on different items, making recommendations based on the similarity between items. Model-based CF algorithm mainly uses the rating information to train corresponding model, and use this model to predict unknown data. These mainly include Bayesian networks [3], clustering models [4], Probabilistic Matrix Factorization [5]. Content-based systems [6] generate recommendations for users based on a description of the item and a profile of the users preference. Hybrid recommendation systems [7] combine both collaborative and content-based approaches, they help improve recommendations that are derived from sparse datasets.

2.2 Deep learning in recommendation

Deep learning is able to efficiently capture unstructured data, such as auditory and visual information, and extract more complex abstractions into higher

[1] www.pandora.com

level data representation [20]. Dieleman et al. [8] proposed a content-based recommender system which used CNN to extract audio signal for Spotify Music. Shumpei et al. [9] presented a RNN based news recommender system for Yahoo News. Covington et al. [10] use historical query, demographic and other contextual information as features, presented a deep neural network based recommendation algorithm for video recommendation on YouTube. Hidasi, Balzs, et al. [11] presented a Session-based recommendations with RNN. Wan, Shengxian, et al. [15] also proposed using RNN to build a next basket recommendation. Tang et al. [23] propose Convolutional Sequence Embedding Recommendation Model to personalized top-N sequential recommendation.

3 Proposed Approach

Our model consists of four layers: embedding, recurrent structure, pooling layer and output layers. We use this model to capture the sequence feature of the user's consumed data. Figure 1 shows the structure of our sequence-based recommender.

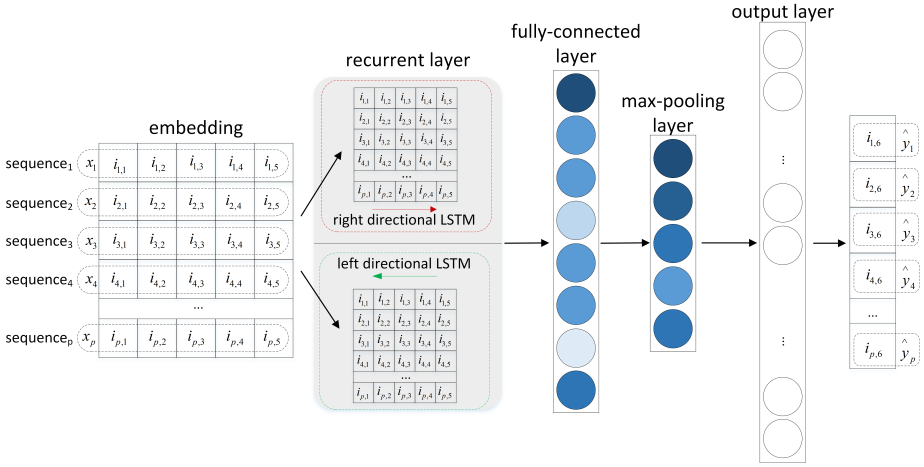

Fig. 1. Framework of sequence-based recommendation

3.1 Notations

Let us assume that $\mathbb{U} = \{u_1, u_2, ..., u_N\}$ is the user set and $\mathbb{I} = \{i_1, i_2, ..., i_M\}$ is the item set. For each user u, there is an observed consumed items sequence $\mathbb{S}_u = \{s_u^1, s_u^2, ..., s_u^{t-1}, s_u^t\}$ in ascending order of time, where s_u^t is the item consumed by user u at time t. The sequential prediction problem is to predict s_u^{t+1} for each user u.

We consider any item can appear only once in the history of a user. Therefore model could recommend items that the user had not yet selected. The output of the network is a softmax layer with a neuron mapping an item in the catalog. We use $p(i_k|S_u, \theta)$ to represent the probability that user u who had a historical consumed items sequence S_u would select item i_k at next time, where θ is the parameters in the network.

3.2 Embedding layer

We use the latest sequences of user consumed items as the features, and the last item as the label, to build a super multiple classification supervised learning model. So in the period of the feature engineering, we need to convert the features into vectors and map them with labels. One-hot vector representation is the most common method to discrete every item. However, One-hot encoded vectors are high-dimensional and sparse. If we use One-hot encoding to deal with 1000 items, each item will be represented by a vector containing 1000 integers, 999 of which are zeros. In a big dataset, this approach is unacceptable considering computational efficiency. Word Embedding shines in the field of Natural Language Processing, instead of ending up with huge One-hot encoded vectors we also can use an embedding matrix to keep the size of each vector much smaller:

$$e(I_i) = EI_i \tag{1}$$

where $E \in \mathbb{R}^{|e|\times|M|}$, $|e|$ is the size of the embedding layer, $|M|$ is the number of items in the training set. So $e(I_i)$ is the embedding of consumed item I_i, which is a dense vector with $|e|$ real value elements.

3.3 User Short-term Interest Learning

We combine a user consumed item I_u^t and other items previous and subsequent I_u^t to present the current interest of user u at time t. The behavior sequences help us to indicate a more precise short-term interest of user. In this recommender, we use a recurrent structure, which is a bidirectional recurrent neural network, to capture the short-term interest of the user.

We define $h_b(I_i)$ as the user's interest before consuming an item I_i and $h_a(I_i)$ as the user's interest after consuming an item I_i. Both $h_b(I_i)$ and $h_a(I_i)$ are dense vectors with $|h|$ real value elements. The interest $h_b(I_i)$ before item I_i is calculated using Equation (1). $W^{(b)}$ is a matrix that transforms the hidden layer (interest) into the next hidden layer. $W^{(cb)}$ is a matrix that is used to combine the interest of the current item with the next item's previous interest. σ is a non-linear activation function. The interest $h_a(I_i)$ after consuming item I_i is calculated in a similar equation. Any user's initial interest uses the same shared parameters $h_b(I_1)$. The subsequent interest of the last item in a user's history share the parameters $h_a(I_n)$.

$$h_b(I_i) = \sigma(W^{(b)}h_b(I_{i-1}) + W^{(cb)}e(I_{i-1})) \tag{2}$$

$$h_a(I_i) = \sigma(W^{(a)}h_a(I_{i+1}) + W^{(ca)}e(I_{i+1})) \tag{3}$$

where the initial interest $h_b(I_1), h_a(I_n) \in \mathbb{R}^{|h|}$, $W^{(b)}, W^{(a)} \in \mathbb{R}^{|h|\times|h|}$, $W^{(cb)}, W^{(ca)} \in \mathbb{R}^{|e|\times|h|}$.

As shown in Equation (2) and (3), the interest vector captures the interest in user's previous and subsequent behavior. We define the temporary status of interest when user taking a behavior I_i as the Equation (4) shown. This manner concatenate the previous temporary status of interest $h_b(I_i)$ before user consuming item I_i, the embedding of behavior I_i consumed item $e(I_i)$, and the subsequent temporary status of interest $h_a(I_i)$ after user consuming item I_i

$$x_i = [h_b(I_i); e(I_i); h_a(I_i)] \tag{4}$$

So using the consumed behavior sequences $\{i_1, i_2, ..., i_{n-1}, i_n\}$, if our model learned the temporary interest status x_{n-1}, users who consumed item i_{n-1} would have a bigger probability to get a recommended item i_n. The recurrent structure can obtain all h_b in a forward scan of the consumed items sequences and h_a in a backward scan of the consumed items sequences. After we obtain the representation x_i of the temporary status of interest when user taking an item I_i, we apply a linear translation together with the $tanh$ activation function to x_i and send the result to the next layer.

$$y_i^{(2)} = tanh(W^{(2)}x_i + b^{(2)}) \tag{5}$$

where $W^{(2)} \in \mathbb{R}^{H\times(|e|+2|h|)}$, $b^{(2)} \in \mathbb{R}^H$ are parameters to be learned, H is the recurrent layer size, $y_i^{(2)}$ is a latent interest vector, in which each interest factor will be analyzed to determine the most useful factor for representing the users consumed items sequences.

3.4 Popularity Trend Learning

When all of the sequences of user's consumed items calculated, we apply a max-pooling layer.

$$y^{(3)} = \max_{i=1}^{n} y_i^{(2)} \tag{6}$$

Max pooling is done by applying a max filter to non-overlapping subregions of the upper representation. With the pooling layer, the number of parameters or weights within the model reduced rapidly, which could reduce the spatial dimension of the upper input volume drastically and lessen the computation cost. We could capture the attribute throughout the entire sequence and find the most popular sequences combination in the whole users' history using the max-pooling layer. The last part of our model is an output layer as follows:

$$y^{(4)} = W^{(4)}y^{(3)} + b^{(4)} \tag{7}$$

where $W^{(4)} \in \mathbb{R}^{O\times H}$, $b^{(4)} \in \mathbb{R}^O$ are parameters to be learned, O is the convolutional layer size.

Finally, a softmax activation function applied to $y^{(4)}$, which can convert the output values to the probabilities of next items.

$$p_i = \frac{e^{y_i^{(4)}}}{\sum_{k=1}^{n} e^{y_k^{(4)}}} \tag{8}$$

3.5 Training

We define all of the parameters to be trained as θ.

$$\theta = \left\{ E, b^{(2)}, b^{(4)}, h_b(B_1), h_a(B_n), W^{(2)}, W^{(4)}, \quad W^{(b)}, W^{(a)}, W^{(b)}, W^{(cb)}, W^{(ca)} \right\} \tag{9}$$

The training target of the network is to minimize the categorical cross entropy loss:

$$\mathcal{L}(y, S, \theta) = -\sum_{u \in \mathbb{U}} [y_u \log p(y_u | S_u, \theta) + (1 - y_u) \log(1 - p(y_u | S_u, \theta))] \tag{10}$$

4 Experiments

4.1 Evaluation and Metrics

As a recommendation model, we will recommend top N item(s) for each user, denoted as $\hat{I}_u^{t+1}$. We adopt $Precision@N$, $Recall@N$ scores to evaluate our model and baseline models. We can define the measures as follows Equations. 11, 12:

$$Precision@N = \frac{\sum_u |\hat{I}_u^{t+1} \cap I_u^{t+1}|}{|\mathbb{U}| * N} \tag{11}$$

$$Recall@N = \frac{\sum_u |\hat{I}_u^{t+1} \cap I_u^{t+1}|}{|\sum_u |I_u^{t+1}||} \tag{12}$$

In order to get a harmonic average of the precise and recall, the $F1@N$ score was measured here, where a higher F1 score reaches, a more effective result gets, which is shown as Eqs. 13:

$$F1@N = \frac{2 \times Precision@N \times Recall@N}{Precision@N + Recall@N} \tag{13}$$

4.2 Dataset and Experimental Settings

Since fewer existing recommendation datasets reserve the continuity of user behavior information, in order to verify our approach is feasible, we perform experiments on a real-world dataset and make it publice: LiveStreaming [2] dataset,

[2] Data is available in https://www.kaggle.com/hailinfu/livestreaming

which collected users' behavior data from a live streaming website in China. Each line in LiveStreaming records a sequence of browsed items of a user in ascending order of time. The initial collected LiveStreaming dataset contains 1806204 lines, which means that contacts 1806204 unique users. The length of each line ranges from 1 to 1060. Total 541772 different items contained in that dataset. We remove those users who are annotated by less than 15 items then randomly select 10 thousand users as the experimental part denoted as LiveStreaming-10M. Finally, 10000 users and 12292 items contain in LiveStreaming-10M. We split 80% of this part into training set, and keep the remaining 20% as the validation set.

Our model is implemented on Keras with TensorFlow-gpu backend, trained on a single GeForce GTX 1050 with 4 GB memory. For hyper-parameter settings, we set: embedding layer size $|e| = 100$, recurrent layer size $H = 200$, convolutional layer size $O = 100$, batch size as 32 and initial number of epochs as 100, learning rate α as 0.01, momentum β as 0.9.

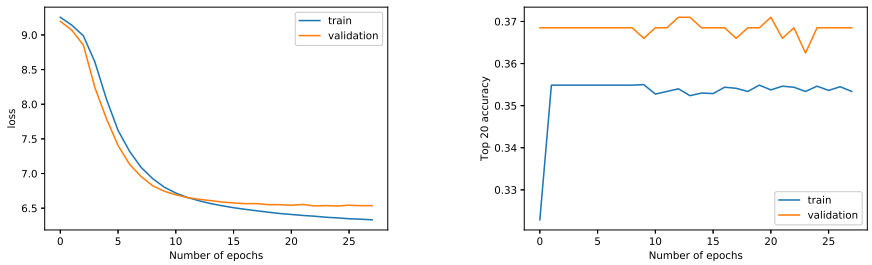

Fig. 2. The training results of CBLSTM model on 80% LiveStreaming-10M with different Time Windows

We also explore how many data the CBLSTM model needs for every user to learn the global sequences tendency and make a good recommendation, so we design a experiment using LiveStreaming-10M with different Time Windows ranges from 1 to 10. When Time Windows equals to 2, means we only use the last item in sequence as label, the latest one item as feature to fit our model.

4.3 Compared Algorithms

In this paper, We compare our model with several existing baselines, and the state-of-the-art approaches in the area of recommendation system:

- **POP**: Popularity predictor that always recommends the most popular items of the training set, it feedbacks the global popularity. Despite its simplicity it is often a strong baseline in certain domains.
- **IBCF**: Collaborative Filtering is one of the most classical method of recommendation, which includes Item-based Collaborative Filtering (IBCF) [2]and

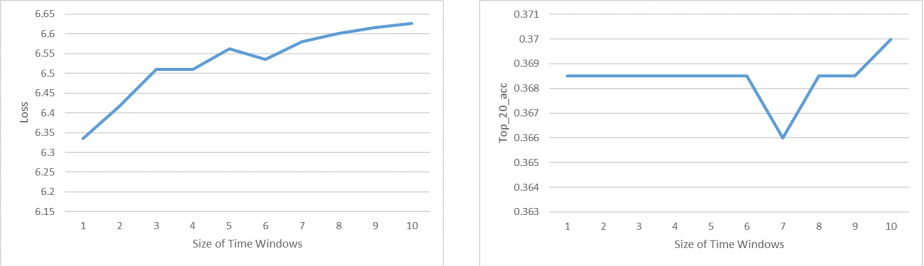

Fig. 3. The training results of CBLSTM model on 80% LiveStreaming-10M with different Time Windows

User-based Collaborative Filtering (UBCF [1], both yet still strong baselines for top-N recommendation. User-based collaborative filtering which evaluate the similarity between users by different users ratings on the item, recommending those items for user consumed by his similar users but not yet consumed by him. However, not every dataset for recommendation contains users' ratings information, in this task, we set the rating 1 if user consumed the corresponding item, or 0 if not. Similarity between u_i and u_j was measured using cosine angle:

$$sim(u_i, u_j) = \cos(\boldsymbol{u_i}, \boldsymbol{u_j}) = \frac{\boldsymbol{u_i} \cdot \boldsymbol{u_j}}{\|\boldsymbol{u_i}\|_2 * \|\boldsymbol{u_j}\|_2} \tag{14}$$

– **UBCF**: User-based Collaborative Filtering which evaluate the similarity between items by different users ratings on the item, recommending items similar to those items consumed already for user. In UBCF, we also use cosine angle to measure similarity between items.

Metrics	POP	UBCF	IBCF	CBLSTM
Precision@1	0.0056	0.0108	0.0112	**0.1135**
Recall@1	0.0056	0.0108	0.0112	**0.1135**
F1@1	0.0056	0.0108	0.0112	**0.1135**
Precision@5	0.00018	**0.0476**	0.00626	0.0463
Recall@5	0.0009	0.0476	0.0313	**0.2315**
F1@5	0.0003	0.0476	0.010433333	**0.07716666666666668**
Precision@20	0.00015	**0.1154**	0.004705	0.018425
Recall@20	0.003	0.1154	0.0941	**0.3685**
F1@20	0.000285714	**0.1154**	0.008961905	0.035095238095238096

Table 1. When the length of recommendation list is 1,5,20 respectively, we compare different approaches using LiveStreaming-10M dataset with Time Windows as 5

Our experiments firstly compare CBLSTM model with itself in different length of sequence. From the Fig.4 we can conclude that our sequence-based

recommendation with CBLSTM also can make recommendations for someone even though with a little information about him. Our approach provides a way to solve the famous cold start problem for recommendation system. Furthermore, from the results (Tab. 1), we can see that the performances of our approach are consistently better than other traditional recommendation baselines on this living broadcast dataset.

5 Results and Conclusion

Overall, in this paper, we propose a novel recommendation framework using generated orders of historical data to predict what users will choose next. Moreover, we introduce convolutional bidirectional Long Short-Term Memory to new application domain: recommendation system. We use embedding matrix to deal with consumed items sequences, and the final model can learn short-term interest of user. Experimental results show that our approach significantly outperforms existing methods, and shows it is suitable to do a short-term prediction.

In future work, we want to use neural network to capture other information not only the sequences of data, and to generate a more accurate and longer term prediction.

References

1. Resnick P, Iacovou N, Suchak M, et al. GroupLens: an open architecture for collaborative filtering of netnews[C]// ACM Conference on Computer Supported Cooperative Work. ACM, 1994:175-186. (1994)
2. Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C]// International Conference on World Wide Web. ACM, 2001:285-295. (2001)
3. Chen YH, George EI. A Bayesian model for collaborative filtering. In: Proc. of the Intl Workshop on Artificial Intelligence and Statistics. (1999)
4. Ungar LH, Foster DP. Clustering methods for collaborative filtering. In: Proc. of the AAAI Workshop on Recommendation Systems. AAAI Press, 1998. 8488 (1998)
5. Salakhutdinov R, Mnih A. Probabilistic matrix factorization[C]// International Conference on Machine Learning. 2008:880-887. (2008)
6. Balabanovi, Marko, Shoham Y. Fab: content-based, collaborative recommendation[J]. Communications of the Acm, 1997, 40(3):66-72. (1997)
7. Burke R. Hybrid Recommender Systems: Survey and Experiments[J]. User Modeling and User-Adapted Interaction, 2002, 12(4):331-370. (2002)
8. Dieleman S, Schrauwen B. Deep content-based music recommendation[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2013:2643-2651.(2013)
9. Okura S, Tagami Y, Ono S, et al. Embedding-based News Recommendation for Millions of Users[C]//Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017: 1933-1942. (2017)
10. Covington P, Adams J, Sargin E. Deep Neural Networks for YouTube Recommendations[C]// ACM Conference on Recommender Systems. ACM, 2016:191-198. (2016)

11. Hidasi, Balzs, et al. "Session-based recommendations with recurrent neural networks." arXiv preprint arXiv:1511.06939 (2015)
12. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. IEEE transactions on knowledge and data engineering, 2005, 17(6): 734-749. (2005)
13. Lai, Siwei, et al. "Recurrent Convolutional Neural Networks for Text Classification." AAAI. Vol. 333. 2015. (2015)
14. Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling[J]. Computer Science, 2014:338-342. (2014)
15. Wan S, Lan Y, Wang P, et al. Next Basket Recommendation with Neural Networks[C]//RecSys Posters. 2015. (2015)
16. Devooght R, Bersini H. Collaborative filtering with recurrent neural networks[J]. arXiv preprint arXiv:1608.07400, 2016. (2016)
17. Yu F, Liu Q, Wu S, et al. A dynamic recurrent model for next basket recommendation[C]//Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016: 729-732. (2016)
18. Ko Y J, Maystre L, Grossglauser M. Collaborative recurrent neural networks for dynamic recommender systems[C]//Asian Conference on Machine Learning. 2016: 366-381. (2016)
19. Jannach D, Ludewig M. When recurrent neural networks meet the neighborhood for session-based recommendation[C]//Proceedings of the Eleventh ACM Conference on Recommender Systems. ACM, 2017: 306-310. (2017)
20. Zhang S, Yao L, Sun A. Deep Learning based Recommender System: A Survey and New Perspectives[J]. arXiv preprint arXiv:1707.07435, 2017. (2017)
21. Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. Journal of machine learning research, 2003, 3(Feb): 1137-1155. (2003)
22. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press. htp://www.deeplearningbook.org.
23. Tang, Jiaxi, and Ke Wang. "Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding." [C]//Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. WSDM 2018: 565-573. (2018).