Sequence-Based Recommendation with Bidirectional LSTM Network

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

School of Computer Science, South China Normal University, Guangzhou 510000, China
{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 novel Sequence-based Recommendation model with Bidirectional Long Short-Term Memory neural network (BiLSTM4Rec) which can capture the sequential feature of items 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 dataset.

Keywords: Recommendation system · Social media data mining
Sequential prediction · Bidirectional recurrent 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 [17] 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"

R. Hong et al. (Eds.): PCM 2018, LNCS 11166, pp. 428–438, 2018.
https://doi.org/10.1007/978-3-030-00764-5_39

(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 found many users consumed music in sequences.

Our work was inspired by the previous study of 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:

- We propose a novel Sequence-based Recommendation model with Bidirectional Long Short-Term Memory neural network, or BiLSTM4Rec for short, which can capture the sequential features of data, as well scales linearly with the number of objectives (both of users and items).
- We regard item sequence as a sentence and use an $M \times d$ embedding matrix E to represent M items which reduces memory cost evidently when faced big data.

2 Related Work

2.1 Traditional Methods in Recommendation

There are three main classes of traditional recommendation systems. Those are collaborative filtering systems, content-based filtering systems, and hybrid recommendation systems [1]. Collaborative filtering (CF) can be generally classified into Memory-based [12,14] CF and Model-based [5,10,16] CF. Memory-based CF [5,12,14,16] systems generate recommendations for users by finding similar user or item groups. Model-based CF systems, *e.g.*, Bayesian networks [5], clustering models [16], Probabilistic Matrix Factorization [10], mainly use the rating information to train corresponding model then use this model to predict unknown data. Content-based systems [3] generate recommendations for users based on a description of the item and a profile of the user's preference. Hybrid recommendation systems [4] combine both collaborative and content-based approaches. However, above traditional methods are not suitable for sequential features capturing. Association rules, *e.g.*, Apriori [2] and FP-Growth [9], can be used to mine sequential features, but they suffer from sensitivity to threshold settings, high time and space complexity, missing of minority rules.

2.2 Deep Learning in Recommendation

Deep learning can efficiently capture unstructured data, and extract more complex abstractions into higher-level data representation [18]. Okura et al. [11] presented a Recurrent Neural Network (RNN) based news recommender system for Yahoo News. Covington et al. [6] used historical query, demographic and

[1] www.pandora.com.

other contextual information as features, presented a deep neural network based recommendation algorithm for video recommendation on YouTube. Hidasi et al. [7] presented a Session-based recommendation with an RNN variant, i.e., GRU. Wan et al. [17] also used RNN to build a next basket recommendation. Zhu et al. [19] used a LSTM variant, i.e., Time-LSTM, to model users' sequential actions. Tang et al. [15] propose Convolutional Sequence Embedding Recommendation Model which incorporates the Convolutional Neural Network (CNN) to learn sequential features, and Latent Factor Model (LFM) to learn user specific features to personalized top-N sequential recommendation. Those works show deep neural networks suit recommendation and RNN architectures are good at modeling the sequential order of objects.

3 Proposed Approach

We propose a novel deep neural model to capture the sequence feature of the user's consumed data. Our model mainly consists of five layers: embedding, recurrent structure, fully-connected layer, pooling layer and output layers. Figure 1 shows the structure of our sequence-based recommender.

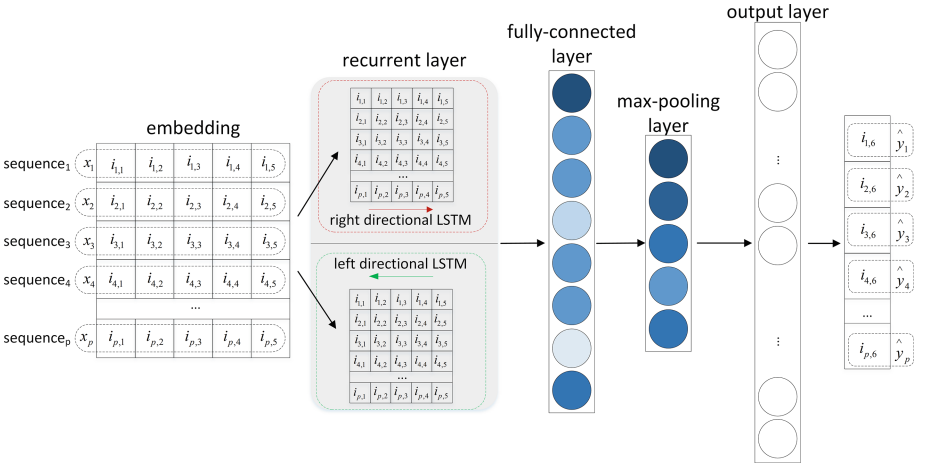

Fig. 1. Framework of sequence-based recommendation

3.1 Notations

Let us assume that $\mathbb{U} = \{u_1, u_2, \ldots, u_N\}$ is the user set and $\mathbb{I} = \{i_1, i_2, \ldots, i_M\}$ is the item set. For each user u, there is an observed consumed items sequence $\mathbb{S}_u = \{s_u^1, s_u^2, \ldots, s_u^{t-1}, s_u^t\}$ 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. Compared with the One-hot encoder (size of $|M| \times |M|$), our item sequence embedding matrix (size of $|e| \times |M|$) reduces memory cost rapidly when dealt with big data.

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 Long Short-Term Memory 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 Eq. (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 Eqs. (2) and (3), the interest vector captures the interest in user's previous and subsequent behavior. We define the temporary status of the interest when a user takes a behavior I_i as the Eq. (4) shown. This manner concatenates 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 $\{i_1, i_2, \ldots, i_{n-1}, i_n\}$, 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 fully-connected 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.

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

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

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

3.6 Time Complexity Analysis

In the embedding layer, it has a matrix multiplication operation with time complexity of $O(n)$. In the recurrent structure, for every sequence, the Bi-directional LSTM structure will apply a forward and a backward scan, and based on the citation [13], the time complexity of the BiLSTM we can know is $O(n)$. The time complexity of the pooling layer and the fully-connected layer is also $O(n)$. The overall model is a cascade of those layers, therefore, our sequence-based recommendation model appears a time complexity of $O(n)$, which is linearly correlated with the number of sequences. The overall time complexity of the model is more acceptable than collaborative filtering ($O(n^2)$) [12,14], so that big data can be effectively processed.

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 following equations:

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

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

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 Eq. (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, to verify our approach is feasible, we perform experiments on a real-world dataset and make it public: LiveStreaming[2] dataset, 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 randomly 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 $H = 200$, fully-connected 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, we adopt classification top 20 accuracy as the evaluation metric. Our code is publicly available[3].

We also explore how many data the BiLSTM4Rec model needs for every user to learn the global sequences tendency and make a good recommendation, so we design an experiment using LiveStreaming-10M with different Time Windows ranges from 1 to 10. When Time Windows equals to 1, means we only use the last item in the sequence as label, the latest one item as feature to fit our model. Figure 3 shows that when Time Windows is less than 6, the results tend to be stable. When Time Windows is too small, much information will lose; too big, it will be hard to train the model. So we choose Time Windows as five in the next experiment.

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:

[2] Data is available in https://www.kaggle.com/hailinfu/livestreaming.
[3] Code is available in https://github.com/fuhailin/BiLSTM4Rec.

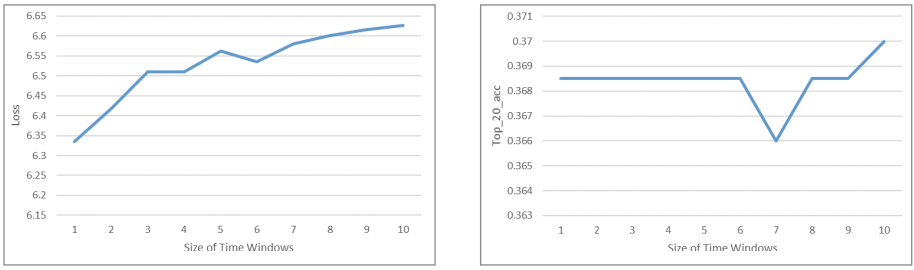

Fig. 2. The training results of BiLSTM4Rec model on 80% LiveStreaming-10M with different Time Windows

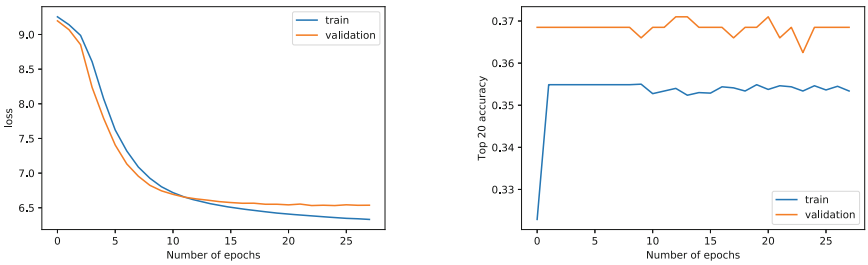

Fig. 3. BiLSTM4Rec training on 80% LiveStreaming-10M with Time Windows as 5, and recommends items with the top 20 probability values

- **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:** As the most classical methods of recommendation, Item-based Collaborative Filtering (IBCF) [14] and User-based Collaborative Filtering (UBCF) [12], both yet still strong baselines for top-N recommendation. 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 evaluates 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.
- **LSTM:** We also compared with the basic Long Short Term Memory (LSTM) [8] model. In the embedding stage, LSTM and BiLSTM4Rec share the same size of the embedding layer. The commonly-used update equations of LSTM are as follows:

$$
\begin{aligned}
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i\right) \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o\right) \\
C_t &= \sigma\left(f_t * C_{t-1} + i_t * \tanh\left(x_t U^g + h_{t-1} W^g\right)\right) \\
h_t &= \tanh(C_t) * o_t
\end{aligned} \tag{15}
$$

Here, we use i, f, o to denote the *input*, $forget$ and *output* gates respectively. We also set the units of LSTM as 200.

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

Metrics	POP	UBCF	IBCF	LSTM	BiLSTM4Rec
Precision@1	0.0056	0.0108	0.0112	0.1135	**0.1165**
Recall@1	0.0056	0.0108	0.0112	0.1135	**0.1165**
F1@1	0.0056	0.0108	0.0112	0.1135	**0.1165**
Precision@5	0.00018	0.0476	0.006223	0.0463	**0.0488**
Recall@5	0.0009	0.0476	0.0313	0.2315	**0.244**
F1@5	0.0003	0.0476	0.010433333	0.077167	**0.081333**
Precision@20	0.00015	**0.1154**	0.004705	0.01825	0.01895
Recall@20	0.003	0.1154	0.0941	0.365	**0.379**
F1@20	0.000285714	**0.1154**	0.008961905	0.034762	0.036095

Our experiments firstly compare BiLSTM4Rec model with itself in different length of sequence. From the Fig. 2 we can conclude that our sequence-based recommendation also can make recommendations for someone even though with a little information about him. Furthermore, from the results (Table 1), we can see that the performances of our approach are better than other traditional recommendation baselines on this living broadcast dataset, especially when faced with short-term predictions. Our experiments have also demonstrated that LSTM is effective for sequence modeling, and our BiLSTM4Rec model does have a certain improvement over the basic LSTM model in the sequence based recommendation.

5 Results and Conclusion

Overall, BiLSTM4Rec is a novel recommendation by modeling recent consumed items as a "sentence" to predict what users will choose next. We introduced the Bidirectional Long Short-Term Memory neural network to a new application domain: recommendation system. We dealt with consumed items sequences by

embedding matrix to save memory cost, and the final model can learn short-term interest of the user. Experimental results show that our approach outperforms existing methods to a great extent, and shows it is suitable to do a short-term prediction. Moreover, it doesn't have an unacceptable time complexity.

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

Acknowledgement. This work was supported by the National Natural Science Foundation of China (No. 61772211, No. 61750110516), and Science and Technology Program of Guangzhou, China (No. 201508010067).

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Springer (2013)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD Record, vol. 22, pp. 207–216. ACM (1993)
3. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
4. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**(4), 331–370 (2002)
5. Chien, Y.H., George, E.I.: A Bayesian model for collaborative filtering. In: AISTATS (1999)
6. Covington, P., Adams, J., Sargin, E.: Deep neural networks for YouTube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 191–198. ACM (2016)
7. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. Comput. Sci. (2015)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
9. Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.Y.: PFP: parallel FP-growth for query recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 107–114. ACM (2008)
10. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, pp. 1257–1264 (2008)
11. Okura, S., Tagami, Y., Ono, S., Tajima, A.: Embedding-based news recommendation for millions of users. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1933–1942. ACM (2017)
12. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW 1994, pp. 175–186. ACM, New York (1994). http://doi.acm.org/10.1145/192844.192905
13. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Computer Science, pp. 338–342 (2014)

14. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295. ACM (2001)
15. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, pp. 565–573. ACM, New York (2018). http://doi.acm.org/10.1145/3159652.3159656
16. Ungar, L.H., Foster, D.P.: Clustering methods for collaborative filtering. In: AAAI Workshop on Recommendation Systems, vol. 1, pp. 114–129 (1998)
17. Wan, S., Lan, Y., Wang, P., Guo, J., Xu, J., Cheng, X.: Next basket recommendation with neural networks. In: RecSys Posters (2015)
18. Zhang, S., Yao, L., Sun, A.: Deep learning based recommender system: a survey and new perspectives. CoRR abs/1707.07435 (2017)
19. Zhu, Y., et al.: What to do next: modeling user behaviors by time-LSTM. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 3602–3608 (2017). https://doi.org/10.24963/ijcai.2017/504