Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 7, Number 1, pages 33–37, January 2018

Capturing Temporal Dynamics of Implicit Feedbacks for Collaborative Filtering by Using Deep Recurrent Neural Networks

Minjia He, Yasuhiko Morimoto
Graduate School of Engineering
Hiroshima University
Kagamiyama 1-7-1, Higashi-Hiroshima 739-8521, Japan
Email: {m162581, morimo}@hiroshima-u.ac.jp

***Abstract*—Collaborative Filtering (CF) is one of successful methods for generating recommendations. However, conventional CF method is not good at capturing sequential behaviors of users. Though it is obvious that users' preference must be affected by experiences of the users in the past. For example, if a user watched a movie and then liked the movie significantly, the user will be interested in movies whose actor/actress/director is the same. In other words, each user's preference towards items is evolving with time dynamically and these temporal dynamics should be noticed. In this paper, we treat users' rating histories as sequential purchasing behavior and utilize such sequential behavior for recommendation task. We use Deep Recurrent Neural Networks (DRNN) to model those purchasing sequences. Experiments on the MovieLens dataset show improvements over previous reported results and demonstrate that our method can utilize users' purchasing behavior data for collaborative filtering while capturing the evolving behaviors and tastes of users better by modeling temporal dynamics implicitly.**

I. Introduction

Since the number of items available in E-Commerce is extremely large, users are always inundated with choices. Recommendation system alleviates this problem by recommending users with items they may be interested. Among recommendation techniques, Collaborative Filtering (CF) [25] is a widely used technique. In order to identify new user-item interactions, CF analyze relationships between users and interdependencies among items. Since a sparse user-item rating matrix is usually used, CF can be regarded as a sparse matrix completion task. In other word, the task is predicting ratings on the missing values' positions. Left part of Figure 1 shows a rating matrix used in CF. Notice that I_i represents an item and U_j represents a user. Rating matrix is a kind of explicit feedback which represents users' preferences towards items explicitly. However, in many applications such explicit feedbacks do not exist. For example, in the situation of E-Commerce website, a large fraction of users are unwilling to give their ratings to items they purchased. In such case, implicit feedback such as purchasing history, click stream, and browsing activity can be utilized for recommendation generation. Right part of Figure 1 shows an example of purchasing matrix. Notice that each 1 represents a purchasing behavior.

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	5	
U_2		2	1	3	1
U_3	3		2	2	
U_4	2	1	5	3	3

	I_1	I_2	I_3	I_4	I_5
U_1	1	0	1	1	0
U_2	0	1	1	1	1
U_3	1	0	1	1	0
U_4	1	1	1	1	1

Fig. 1. Rating matrix and Purchasing matrix.

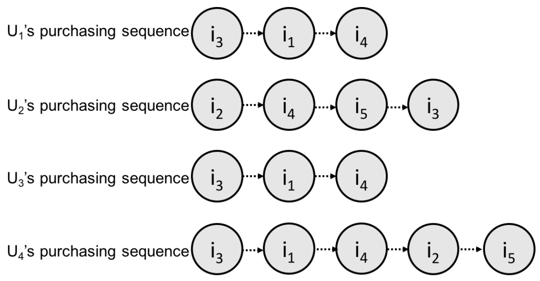
U_1's purchasing sequence $i_3 \to i_1 \to i_4$
U_2's purchasing sequence $i_2 \to i_4 \to i_5 \to i_3$
U_3's purchasing sequence $i_3 \to i_1 \to i_4$
U_4's purchasing sequence $i_3 \to i_1 \to i_4 \to i_2 \to i_5$

Fig. 2. Purchasing sequences generated from purchasing matrix.

Users' preferences for items are evolving over time, we call this temporal dynamics. For example, users' tastes are evolving dynamically as new items emerge. As stated in [13], the analysis of such time drifting data needs to find the right balance between discounting temporary effects that have very low impact on future behavior, while capturing longer-term trends that reflect the inherent nature of the data. They modeled temporal dynamics of users' preferences as time drifting parameters and combined them with latent factor model [14] in CF. Although it is useful, this approach is limited by its simplicity and is defective in capturing temporal dynamics.

In this paper, we focus on generating personalized recommendations by utilizing users' purchasing histories. Based on the purpose of capturing evolving preferences of users, we treat each user's purchasing history as a sequence of purchased items sorted by timestamp. Figure 2 shows a set of purchasing sequences generated from purchasing matrix data in Figure 1. Notice that in this paper, we only consider purchasing behaviors of users and assume that all rating values on purchased items are unavailable. Therefore, in experiments, we transform rating matrix into purchasing history matrix by simply regarding rating values as 1s. This is reasonable because the rating matrix dataset does not only tell us the rating values, but also which movies users watched, regardless of how they rated these movies.

Given a purchasing sequence of a user, we are interested in predicting the most possible next element of this sequence. The motivation of doing such prediction is based on the idea that users' past purchasing histories have influence on their future behaviors. The closer a purchasing history is, the larger the influence for next purchase has. In the meanwhile, we assume that similar users will have similar purchasing sequential patterns, which is similar to the assumption in CF. As a result, the collaborative filtering recommendation task becomes a sequence prediction task.

Recently, a number of tasks have been done using Recurrent Neural Networks (RNNs), such as image caption generation [28], speech recognition [9] and language modeling [17]. RNNs have shown a great success at modeling those sequential data. For handling the sequential prediction task in recommendation field, we choose Deep Recurrent Neural Networks (DRNNs) as our model because it shows a great ability in long-short term memory which is useful for modeling temporal dynamics. By modeling purchasing history, we are not only utilizing sequential patterns from all users to generate recommendations in the manner like CF, but also are monitoring the changes of users' preferences. Notice that the collaborative filtering task is done implicitly in our model because we are modeling sequential patterns generated from all users' purchasing historires data. Contrast to approach in [13], our model can learn temporal dynamics without setting special designed parameters.

The rest of this paper is organized as follows. Section II reviews related works. Section III presents the definition and notations used in RNNs. We explain the details about our proposed model in Section IV. Then, we examine the proposed model by experiments in Section V. After that, Section VII concludes the paper.

II. Related Work

A. Collaborative Filtering

Most recommendation systems are based on Collaborative Filtering. There are two primary approaches in CF, one is neighborhood approach, another is latent factor approach. Neighborhood models are based on the similarity among users or items. For instance, two users are similar if they have rated a similar set of items, which is called user-based approach [3]. On the other hand, item-based approach [23] compute a user's preference for an item based on user's own ratings on similar items. Latent factor models project users and items as vectors in the same latent-factor space. In such space, users and items are directly comparable. Most latent factor models are based on factoring the user-item rating matrix using Singular Value Decomposition (SVD) [20] .

B. Neural Networks in Recommendation Systems.

In [22], Restricted Boltzmann Machine (RBM) is used for handling the recommendation task. Extending the idea of this work, in [24], completion of rating matrix is split into encoding step and decoding step. This is a kind of latent factor model and the main task is the same as the one in Collaborative Filtering. Latent factors are extracted by the neural network model called Stacked Denoising Autoencoder (SDA). In [15], RNNs are used to model users' sequential tweets for purchasing behavior prediction. It is believed that previous tweets have influence on following tweets and tweet is a efficient indicator of user's future purchasing behavior. RNN is also chose as one of the approach in [2] for quote recommendation. This work focuses on recommending proverbs and famous statements to user who are writing based on the current body of the text.

C. Session-based Recommendation Systems

In [8], RNNs are used for session-based recommendations. A session contains pages clicked by a user in a web session. Each session is regarded as a sequence of pages sorted by timstamp. The target is to generate the real-time recommendation based on session data rather than historical rating data. Session-based approach focus on recommending users with items they are searching for currently. As a result, the path of pages needed for each user to click is shorten, thus time is saved. This work uses DRNN with multiple hidden layers to model session data. Since length of sessions are very different and the goal is to capture how a session evolves over time, the model is trained in a session-parallel manner. Another important reason for using session-parallel training manner is that sessions are time-sensitive, which means each session is generated by a user in a short period of time, thus breaking down a session into fragments will make no sense. During training stage, losses are measured using ranking loss such as Bayesian Personalized Ranking (BPR) [21] and TOP1 [8].

The work in [8] is related to ours, but instead of modeling sessions, we build a DRNN-based model to deal with users' purchasing history data. We train our DRNN model in a sequence-to-sequence manner [17] for several reasons. Firstly, in our case, purchasing sequences are not so time-sensitive as sessions. Secondly, purchasing sequences are in average much longer than session sequences. There are even lots of purchasing sequences with length larger than one thousand. Finally, our goal is to capture the temporal dynamics within purchasing sequences while executing collaborative filtering. This training manner enables our model to generate prediction based on a fragment of one purchasing sequence rather than the whole sequence. As for measuring loss, instead of using ranking loss, we use cross entropy to measure the losses between predictions and true labels.

III. Recurrent Neural Networks

A. Basic Recurrent Neural Network

RNN is a kind of artificial neural network good at modeling sequential data. It maintains a state which can be updated as time goes on, so previous information can be memorized

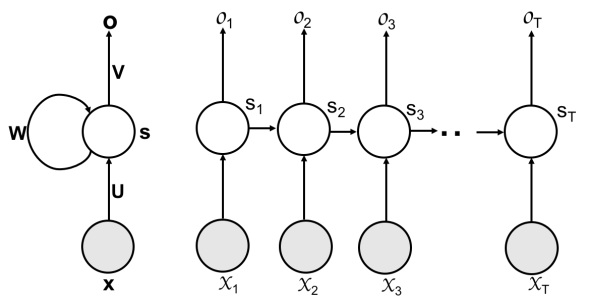

Fig. 3. Basic RNN structure and unfolded RNN structure.

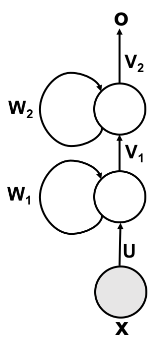

Fig. 4. DRNN with two hidden layers

and utilized for current prediction. Each sequence is split into multiple elements. At each time step, one element is used as an input. As shown in Figure 3, blank circle represents a state and gray circle represents a input. If we unfolded the RNN according to time steps, we can represent RNN as the network like the one shown in right part of Figure 3. Output and input at each time step is denoted as o_i and x_i. The state s_i is updated based on previous s_{i-1} and current input x_i:

$$s_i = f(Ux_i + Ws_{i-1})$$

where f is a non-linear activation function such as ReLU, tanh and sigmoid.

The output at time step *i* is computed as:

$$o_i = softmax(Vs_i)$$

B. Deep Recurrent Neural Network

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [16]. It has attracted lots of attentions and shown a great success in the field such as speech recognition [9], image recognition [7] and natural language processing [26].

Since RNN has self-loop structure, it is already a kind of deep neural network to some extent. However, [18] shows that RNNs can be made deeper in several different ways. Their experiments demonstrate that DRNNs can learn representations of data better and outperform vanilla RNNs in lots of tasks. In this paper we make RNN deeper by adding multiple hidden layers. The structure of our model in shown in Figure 4.

IV. Proposed Approach

A. RNNs for Recommendations

By sorting according to timestamp, each user's purchasing history can be represented as $[x_1, x_2, ..., x_{T-1}, x_T]$, where x_i (1 $\leq$ i $\leq$ T) is the index of one rated item from the items set of size m. The structure of our model is shown in Figure4. In our model, we use m-dimensional one-hot vector to represent x_i, which means $x_i \in \mathbb{R}^m$. We call such sorted purchasing histories as purchasing sequences.

Given a purchasing sequence **x** = $[x_1, x_2, ..., x_{r-1}, x_r]$ (1 $\leq$ r $\leq$ T), our DRNN model will give the output **y**, where **y** = $[y_1, y_2, ..., y_{m-1}, y_m] \in \mathbb{R}^m$. **y** is a probability distribution

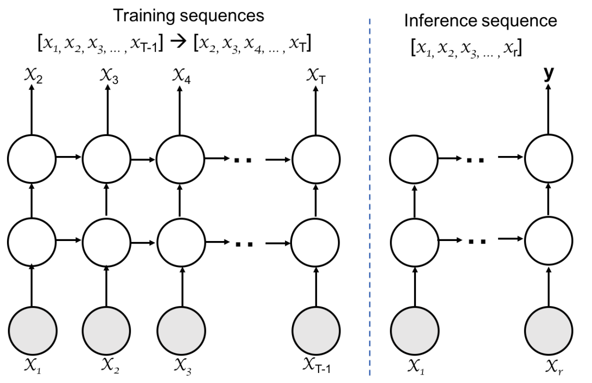
Training sequences
$[x_1, x_2, x_3, ..., x_{T-1}] \rightarrow [x_2, x_3, x_4, ..., x_T]$
Inference sequence
$[x_1, x_2, x_3, ..., x_r]$
x_2
x_3
x_4
x_T
y
x_1
x_2
x_3
x_{T-1}
x_1
x_r

Fig. 5. Training and test stages using RNN with two hidden layers.

over all m items, where y_i is the probability of item *i* to be the next element of this sequence. Therefore, recommendation task is changed into a sequence predication problem. Besides, since recommendation systems usually recommend multiple items to each user, we rank **y**'s elements and choose the top-*K* items as the recommendation list.

B. Training Stage

During the training stage, as shown in Figure 5, we set a fixed size of time steps T for unfolding the network. We use $\{[x_1, x_2, ..., x_{T-1}]_j, [x_2, x_3, ..., x_{T-1}, x_T]_j\}_{j=1}^{N}$ as training data and train the model in a sequence-to-sequence manner. $[x_1, x_2, ..., x_{T-1}]_j$ is the input and $[x_2, x_3, ..., x_{T-1}, x_T]_j$ is the corresponding true output. Backpropagation-Through-Time (BPTT) [27] is used for propagating gradients of errors. Error is computed using cross entropy loss function:

$$cross\_entropy(\mathbf{y}', \mathbf{y}) = -\sum_{i=0}^{n} y_i' log(y_i)$$

(Where $\mathbf{y}'$ is the true probability distribution and **y** is the predicted probability distribution.) BPTT is a variant of backpropagation method used in training feedforward neural networks. Using BPTT, error at each time step is backpropagated to previous time steps. i.e., BPTT applies backpropagation method to unfolded RNN. Besides we use Adagrad [6] as the optimization method. It is a modified stochastic gradient descent method with per-parameter learning rate. Using Adagrad, weights are updated in a mini-batch manner, which means at each updating step, gradient of a random sampled mini-batch is computed rather than computing gradient of the whole training data. In addition, dropout [29] is used in our model as regularization method for alleviating overfitting.

C. Inference Stage

Figure 5 shows how the inference works. The goal of inference is to predict the next element based on the information provided by an inference sequence with length r. Each inference sequence is a segment at the end of a purchasing sequence. i.e, each user's recent purchasing history is used for generating recommendations. By generating inference sequence in this way, the bad impact of staled purchasing history is avoided. Besides, our model is trained in a sequence-to-sequence manner, when it comes to inference, input purchasing sequences may take different lengths. Therefore, the value of r in Figure 5 is variable for different purchasing sequences. Besides, as stated in [10], it is difficult for RNNs to capture long-term dependencies within sequence very well. As a result, the length of inference sequence used is limited by the dependencies learning ability of the model.

Since recommend a list of items is the common choice in recommendation systems, we ranked items in the prediction **y** by their values and choose the top-*k* items to form the recommendation list.

V. Experiment

A. Experimental Setup

We use Movielens-1M dataset in our experiments. This dataset contains 1,000,209 ratings over 3706 movies and 6040 users. Dataset is splitted into 3 subsets for training, validation and testing usages. We generate these subsets by dividing users into 3 groups. As a result, 3040 users' data are used as training data and 1500 users' data are used for validation and test respectively.

The model contains 2 hidden layers and each layer contains 200 Gated Recurrent Unit (GRU) [5] cells. We use GRU since it achieves comparable performance as Long Short Term Memory (LSTM) [11] while using less parameters. Our model is trained with 50 time steps using BPTT. i.e., inference sequence can not be longer than 50. Training sequence with length less than 50 is padded with zeros at the end of the sequence. Besides, errors of paddings are masked out. The initial learning rate of Adagrad method is set to 1.0 and mini-batch size is set to 100. To alleviate the gradient expoloding problem [19], we clip the norm of the gradients (normalized by minibatch size) at 5. Dropout probability is set to 0.5. The model is implemented using Tensorflow [1], which is a deep learning platform developed by Google. A GeForce GTX 1080Ti GPU is used to boost the training speed.

TABLE I
Comparison of top-K recommendation methods. (K=10)

Method	Precision(%)	Recall(%)	F1-Score
Item-based	10.83	3.13	5.23
User-based	12.12	3.50	5.43
SVD	22.44	6.48	10.06
MF-implicit	18.58	5.37	8.33
RNN	26.46	7.65	11.87

B. Experimental Results

We compare several recommendation methods with our approach. Those methods are Item-based CF, User-based CF, SVD and Matrix Factorization for implicit feedback (MF-implicit) [12]. Among these methods, MF-implicit and our DRNN-based method use purchasing history data while others use users' ratings for generating recommendations. Notice that MF-implicit is the state-of-the-art work on collaborative filtering using implicit feedback dataset.

Evaluations are executed using Precision, Recall and F1-Score measurements. Table I shows the evaluation results of five methods on top-K recommendation task where K equals 10. It demonstrates that our DRNN-based approach outperforms other four approaches.

To clarify the ability of our approach, as shown in Figure 6 and Figure 7, we further measure recalls when changing the value of K and measure the recall-precision relationship. Notice that we use dash lines for methods utilizing implicit feedback data and use solid lines for methods utilizing ratings data. From Figure 6, we observe that when K is less than 15, our RNN-based model outperforms all other approaches on the recall measurement. A related phenomenon also appears in Figure 7. We observe that when the recall is lower than 20%, RNN-based approach outperforms other approaches on precision. This an evidence showing that users' temporal dynamics are well catched by DRNN-based model and global sequential patterns are useful in handling the collaborative filteirng task. Besides, the rapid decrease of precision in Figure 7 is in accord with the feature of RNN stated in [4]. Since it is difficult for RNN to learn long-term dependencies, RNN shows a rapid down in ability for generating large-size recommendation list. However, since our model outperforms others in cases where size of K is less than 15, it is suffice to be applied to recommendation situations such as movie recommendations and E-Commerce websites.

VI. Conclusion

In this paper, we transformed the recommendation task into a sequence prediction task by utilizing RNN models. In our method, we explored users' purchasing history data in a new way. Patterns within users' rating sequences are well modeled by the DRNN model with millions of tunable parameters. Based on experiments, we observe a 1.17%, 4.02% and 1.81 improvement over previous methods on Recall@10, Precision@10 and F1-Score. Therefore, we conclude that our method is competitive with previously proposed methods for

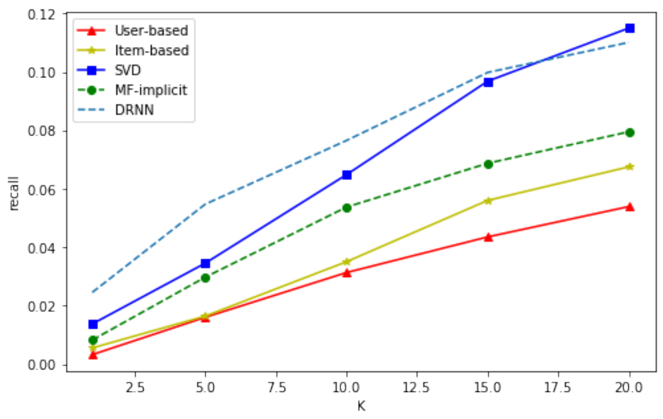

Fig. 6. Recall at K.

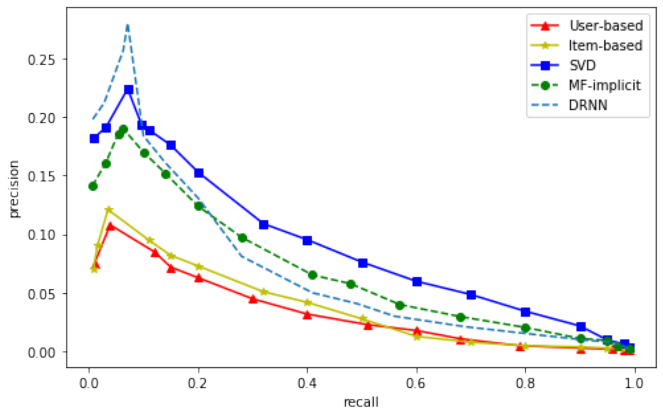

Fig. 7. Recall versus Precision.

handling recommendation using implicit feedback data. Besides our model keeps the advantage of collaborative filteirng and can capturing temporal dynamics by utilizing patterns within sequences.

Acknowledgment

This work is supported by KAKENHI (16K00155,17H01823) Japan.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Yeonchan Ahn, Hanbit Lee, Heesik Jeon, Seungdo Ha, and Sang-goo Lee. Quote recommendation for dialogs and writings. In *CBRecSys@ RecSys*, pages 39–42, 2016.

[3] Robert M Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 7–14. sn, 2007.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[9] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[13] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[15] Mandy Korpusik, Shigeyuki Sakaki, Francine Chen, and Yan-Ying Chen. Recurrent neural networks for customer purchase prediction on twitter. In *CBRecSys@ RecSys*, pages 47–50, 2016.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[17] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[18] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[19] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[20] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[22] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[23] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[24] Florian Strub and Jeremie Mary. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS Workshop on Machine Learning for eCommerce*, 2015.

[25] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[27] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[28] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[29] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.