IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 11, 2021, accepted April 28, 2021, date of publication May 13, 2021, date of current version May 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3079922

Recommendation System With Hierarchical Recurrent Neural Network for Long-Term Time Series

BYEONGJIN CHOE, TAEGWAN KANG, AND KYOMIN JUNG, (Member, IEEE)
Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea

Corresponding author: Kyomin Jung (kjung@snu.ac.kr)

This work was supported in part by the Brain Korea 21 (BK21) FOUR Program of the Education and Research Program for Future Information and Communication Technology (ICT) Pioneers, Seoul National University, in 2021, and in part by the Automation and Systems Research Institute (ASRI), Seoul National University.

ABSTRACT Nowadays, recommendation systems are widely used on various services. This system predicts what item a user will use next using large amounts of stored user history. Recommendation systems are commonly applied in various fields as movies, e-commerce, and social services. However, previous researches on recommendation systems commonly overlooked the importance of item usage sequence and time intervals of the time series data from users. We provide a novel recommendation system that incorporates these temporal properties of the user history. We design a recurrent neural network (RNN) model with a hierarchical structure so that the sequence and time intervals of the user's item usage history can be considered. The model is divided into two layers: a layer for a long time and a layer for a short time. We conduct experiments on real-world data such as Movielens and Steam datasets, which has a long-time range, and show that our new model outperforms the previous widely used recommendation methods, including RNN-based models. We also conduct experiments to find out the influence of the length and time interval of sequences in our model. These experimental results show that both sequence length and time interval are influential, indicating that it is important to consider the temporal properties for long-term sequences.

INDEX TERMS Machine learning, recommender systems, recurrent neural networks.

I. INTRODUCTION

Recently, recommendation systems have become widespread on many websites, including e-commerce. Based on the user information, a recommendation system suggests items that the users might want to buy. Especially, customizing a recommendation system to the users is a key to their efficacy. Various recommendation systems have been proposed to predict users' behavior and make better recommendations. Generally, recommendation systems predict users' personalized preferences and find items with the highest preference based on their past usage records.

In a recommendation system, temporal information in the users' past usage record is one of the important elements. Items recently purchased and items purchased long ago have different influences on current users' preferences. If someone who used to enjoy horror movies in the past has recently watched romance movies, he or she is now more likely to prefer romance movies to horror movies. Order information is also part of temporal information. For example, people usually buy a keyboard after buying a computer, but it will be rare to buy a computer after buying a keyboard.

However, the importance of temporal information has not been carefully considered in previous recommendation systems. Classical recommendation methods [1]–[3], such as content-based filtering and collaboration filtering, fail to take advantage of temporal information, looking at all records equally. Recently, several studies have been conducted using order information to make use of temporal information. Hidasi *et al.* [4], [5] suggest a recurrent neural network (RNN)-based recommendation model, called the Session-based RNN recommendation system. RNN is a well-known model for processing sequential data and is widely applied to the language model and speech recognition. Session-based RNN uses implicit feedback – click, visit, and any other session information – as input sequence of RNN.

The associate editor coordinating the review of this manuscript and approving it for publication was Nilanjan Dey.

However, many previous studies lacked attempts to utilize the time interval between the use of items. In a short sequence, the time interval between item use may not be an important factor, but in a sequence over a long period of time, the time interval becomes more important. In many previous studies, there is no difference between watching the two films one year apart and one day apart. Moreover, a week ago for a user who watched a movie every day for a week, will be much further away in order than a year ago for a user who watches a movie only once a year.

In this study, we present an RNN-based hierarchical model to consider both long sequence lengths and time interval information. A typical RNN-based model struggles to differentiate between events that occur at a certain interval of time and events that occur irregularly. Our model applied a hierarchical structure to effectively use time information to handle different time intervals differently in the model.

Our model has two layers: a short-term layer and a long-term layer. The short-term layer deals with short-term sequences that were commonly dealt with in previous models. In this paper, we use a basic RNN structure. Our model views the entire sequence as a bunch of short-term sequences. The long-term layers remember information from previous short-term sequences and deliver it over a long time considering the time interval.

We did experiments on datasets of Movielens with 1 million movie reviews and Steam. These datasets deals with several years of time and is, therefore, suitable for experiments on long-term time series. Experimental results show our model outperforms previous RNN models considering sequential order as well as models that do not consider sequential order. To further analyze our model, we conducted an experiment on sequence length and the influence of a time interval. In the experiment on sequence length, our model does not show impressive performance compared to baselines when the sequence length is very short, but is powerful when sequence length is long. We also conducted an ablation study on how much time interval should be considered between items. This showed that when the time interval is long, it is helpful to moderately weaken the influence of the item whose order in the item usage sequence is most recent. This shows why the model presented in this study is particularly strong in long-term sequential datasets.

II. RELATED WORKS

In this session, we introduce a number of past studies related to the recommendation system. This covers from recommendation system studies where sequence is not considered to recent RNN-based studies.

Collaborative Filtering (CF) is the general recommendation system method [1]. CF is based on the notion that people who have the same preferences might choose similar items. CF analyzes a user-item matrix to look for similar neighbors of users and then proposes items through these neighbors. However, item-to-item CF focuses on item–item matrices. The idea behind item-to-item CF is that items that have similar properties might be chosen by the same people. One of the other CF methods is Content-Boosted Collaborative Filtering (CBCF) [6], which uses ratings as a vector form directly.

Matrix Factorization (MF) gives insight to methods that utilize item ratings. MF maps user-latent factors and item-latent factors to joint dimensions. The goal of MF is to ensure that the product of these latent factors converges on the rating. The pure singular-value-decomposition-based (PureSVD) matrix factorization method and weighted regularized matrix factorization (WRMF) are examples of MF methods. MF is also used for recommendations with implicit feedback. BPR [3] is applied to MF to provide item prediction from implicit positive-only feedback. This method formulates the recommendation problem as a ranking problem. In addition, some BPR-based recommendation systems try to add social network information to basic features [7], and some BPR-based systems grade items to consider user preferences [8].

Non-sequential Neural Networks are also used in recommendation systems. Most of the deep-learning models used in recommendation systems are based on CF or MF, because deep-learning models can find more precise similarities in CF and latent factors in MF. The restricted Boltzmann machine (RBM) exhibits good performance in model phoneme recognition [9] and the classification problem [10], because it can easily obtain accurate features. Similarly, the RBM can be applied to CF-based recommendation systems. The RBM automatically extracts similarities or latent factors for items. Auto-encoders-based recommendation systems also have strengths in finding associations in features. The stacked denoising auto encoder (SDAE) [11] analyzes relationships among items and can obtain more accurate relations than the general CF model. collaborative deep learning (CDL) [12] jointly performs encoding and rating considering MF processes with feedforward networks. The collaborative denoising auto encoder (CDAE) [13] matches with top-N recommender systems using an auto encoder. NeuMF [14] combines a MF-based method and a multi-layer perceptron-based model to predict the item that will be used by a particular user. However, since the number of users affects the input of the model, there is a problem that if a new user comes in, a whole new learning is required. The key to these deep-learning models is that they can easily extract better latent factors than non-deep-learning models. However, since these models only change the existing CF/MF models into a deep-learning form, they still have limitations such as failure to consider temporal properties.

Recurrent Neural Networks (RNNs) work well with sequential data. RNNs are generally applied to language modelsfancyhdr [15] and speech recognition [16]. Hidasi *et al.* suggested an RNN-based recommendation model called a session-based RNN recommendation system [4]. The session-based RNN uses only implicit feedbacks – clicks or watch history – as the RNN's input sequence. Based on the implicit feedbacks, the session based RNN

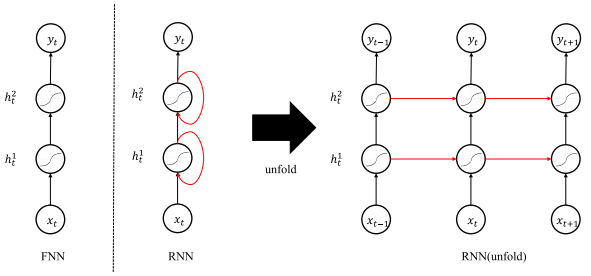


FIGURE 1. Illustration of general Feedforward Neural Network (FNN) architecture and Recurrent Neural Network (RNN).

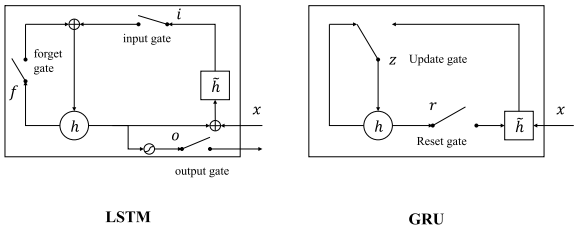


FIGURE 2. Illustration of Long Short Term Memory (LSTM, left) and Gated Recurrent Unit (GRU, right): f, i and o are respectively the forget gate, input gate, and output gate. h and $\tilde{h}$ are the hidden (cell) state and the candidate of hidden (cell) state in LSTM. In one LSTM cell, f, i, and o must be trained simultaneously, so LSTM must have too many parameters. However, x is the input in GRU. z denotes the update gate and h denotes the reset gate, there are only a few gates. h and $\tilde{h}$ are respectively the hidden state and the candidate hidden state. In GRU, the hidden state and output are the same. In this case, h is the hidden state and also the output of the GRU.

learned to rate latent scores of the items like BPR. Wu *et al.* proposed an RNN-based recommendation system [17] that exploits user-profile data compounded with session information data and guesses the items users want.

III. PRELIMINARIES

A. RECOMMENDATION SYSTEM

The goal of a recommendation system is to propose the items that the users are most likely to prefer to use. In the recommendation problem that we want to solve, an event is a user's use of a specific item. One event contains information about which users used which items and when, which is a tuple of (user, item, timestamp). Each user has multiple events with different timestamps, and listing them in temporal order generates a sequence of events. If a sequence of events $S = \{x_1, \ldots, x_n\}$ is given for any user, the recommendation system finds an item of the next incoming event x_{n+1} by using given sequence S. More specifically, the recommendation system computes the probabilities of each item used in the next event and makes a ranked list of the probabilities that are sorted in ascending order for each user.

It is a very difficult problem for users to pinpoint which item to use next among so many items. In addition, it is also very useful to present multiple strong candidates to users rather than a single answer in the practical recommendation system application. Therefore, to increase convenience and efficiency, recommendation systems that propose the top-N number of ranked items in the ranked list to the user, referred to as top-N recommendation systems, are widely used. For evaluating the model in a test phase, the researchers compare the ranked list with the actually used items in the test set.

B. RECURRENT NEURAL NETWORK

As described in Sections 1 and 2, the RNN works well in many sequential data analysis tasks. In this section, we describe the architecture of the RNN. The RNN is a modified version of a feedforward neural network. In Fig 1, the general feedforward network (left) assumes that all the inputs are independent from each other, but the hidden layer of the RNN is dependent on previous information caused by the red-colored edges. These red-colored edges are called ''recurrent edges,'' and they start from the hidden layers and come in the same hidden layers. Given the input dataset, the RNN can renew the hidden unit state every time using the past hidden unit state (recurrent) and new input. In this manner, the RNN can induce time-dependency on sequential data. In summary, the formulas updating the hidden state h and output y in the RNN structure are as follows:

$$h_t = f(U_t x_t + W_t h_{t-1} + b_h),$$
$$y_t = g(W_y h_t + b_y)$$

where the weight parameters are given by U and W, and the bias parameters are given by b (Fig 1).

However, this basic RNN structure has a practical problem referred to as the vanishing gradient problem [18]. The LSTM [19] is a particularly popular structure in the RNN-based deep-learning architectures that solves the vanishing gradient problem. Cho *et al.* suggested the GRU [20], which simplifies the LSTM by using reset gates and update gates that are similar to the switch structure. The reset gate determines how to combine previous hidden state information with input information, and the update gate determines the extent to which the next hidden state reflects the previous hidden state information and new input information. Formulas of the GRU are as follow:

$$z = \sigma(x_t U^z + s_{(t-1)} W^z)$$
$$r = \sigma(x_t U^r + s_{(t-1)} W^r)$$
$$h = tanh(x_t U^h) + (s(t-1) \cdot r) W^h)$$
$$s_t = (1 - z) \cdot h + z \cdot s(t-1)$$

where the reset gate is given by r, the update gate is given by z, the activation function is given by σ and the state vector is given by s (Fig 2). These structures are currently widely used in practice, and this study also builds models using GRU.

IV. MODEL DESCRIPTION

In this section, we explain our new recommended system model for long-term time-series data, *Hi-RNN*. We describe how our model can handle the long-term time series data with various time intervals.

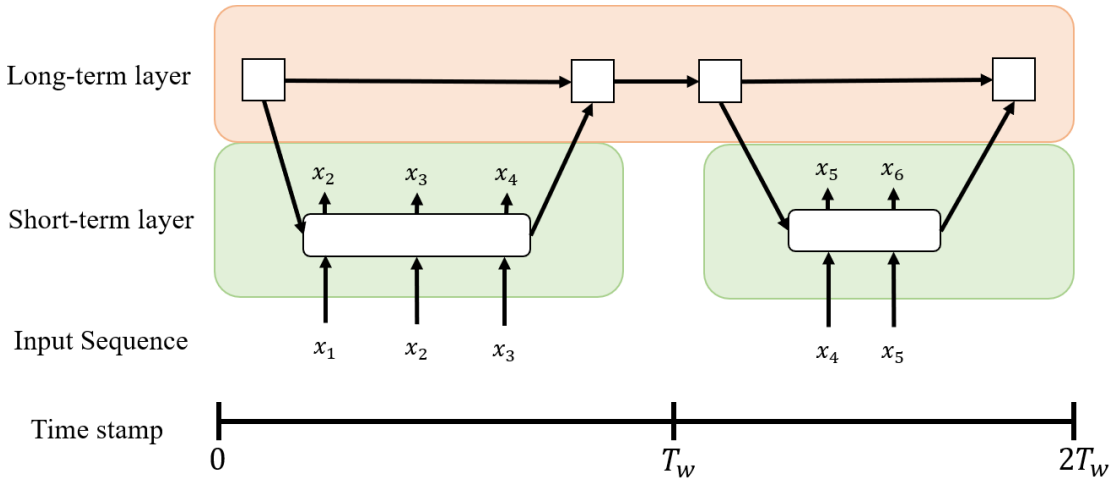


FIGURE 3. This is a figure of our model with a hierarchical structure. Upper square nodes are the long-term layer, and the lower rounded-square nodes are the short-term layer. The short-term layer receives input whenever an event occurs.

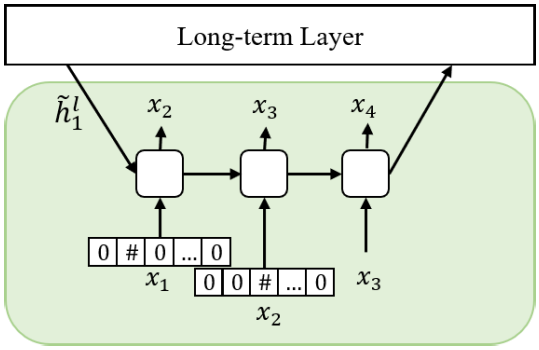


FIGURE 4. This is a figure of one short-term layer block in hi-RNN.

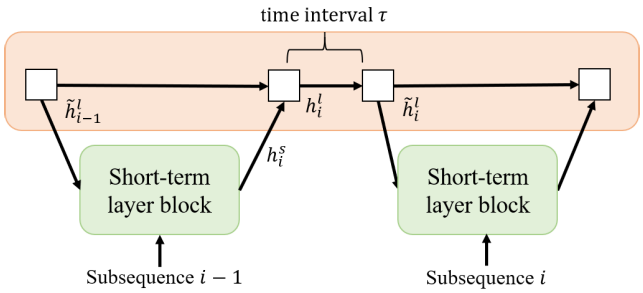


FIGURE 5. This is a figure of a long-term layer in hi-RNN.

The RNN structure described in Section 3.2 is a neural network specialized in dealing with sequential data. However, as shown above, the basic RNN structure lacks consideration of the interval between inputs. If time intervals between item usage events are all the same, we can ignore the time interval information and use basic RNN. However, if time intervals are irregular, the basic RNN cannot handle them accurately. In addition, the long-term sequence data we use can vary in time intervals from minutes to years. Therefore, we constructed a hierarchical model consisting of two layers: *short-term* and *long-term*, which solves this problem.

(Short-term Layer) A short-term layer is a layer for events within a short period. Our model deals with sequence $S = \{i_1, \ldots, i_n\}$ divided into T_w intervals. If the time stamp of i_1 is 0, the first sub-sequence S_1 contains item usage records(events) from 0 to T_w. In this way, the whole sequence is divided by sub-sequences $\{S_1, \ldots, S_m\}$, where $m = \lceil T_n/T_w \rceil$ is the number of sub-sequences.

The short-term layer consists of small basic RNN structures that receive a sub-sequence as input. The time interval between events within the same sub-sequence is very short compared to the overall data length. Because of the short time interval, the influence of the time interval is significantly reduced, and only the sequential order of the inputs is sufficient. Thus, the RNN structure that does not consider time intervals can also sufficiently handle events within a short period.

Prediction of the next item is made in the short-term layer. To predict the next item, the basic RNN uses item information of the current time and hidden state up to the previous time. In the short-term layer, the current state of the long-term layer h_{t-1}^l is additionally brought in to get item usage information from the distant past.

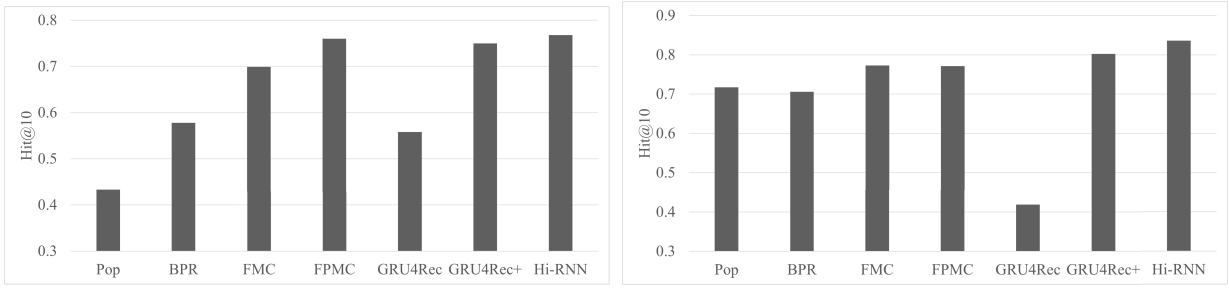
$$h_t^s = f(U^s x_t + W^s h_{(t-1)}^s + W^l h_{(t-1)}^l + b^s)$$

where x_t is the vector in which the item information of the event number t is converted into one-hot encoding.

The last node's hidden state is used in long-term layers as a summary of the current sub-sequence. In other words, m hidden states can be obtained from m sub-sequences through the short-term layer.

(Long-term Layer) The long-term layer transmits past information to the short-term layer block by using an RNN-based structure. The long-term layer receives m last hidden states of the m short-term layer blocks as follows:

$$h_t^l = f(U^l h_t^s + V^l h_{(t-1)}^l + b^l)$$



(a) Movielens1M (b) Steam

FIGURE 6. Hit@10 for movielens1M dataset (left) and steam dataset (right) using the baseline methods and our method (Hi-RNN).

where h_t^s is a short-term layer's hidden state and $h_{(t-1)}^l$ is a long-term layer's previous hidden state. These m inputs are a sequence with a constant time interval of T_w. That is, the short-term layer transforms irregular time interval data to equal time interval data. Therefore, the long-term layer can process data without losing information about time intervals, even if they are made up of an RNN structure. The long-term layer also reduces the problem of RNNs, which are more likely to lose historical information as the input length increases, because multiple inputs are grouped and the length of the entire input is shorter.

However, depending on the time distribution of events in the sequence, there may be an empty subsequence. In this case, there is no input on the node in the long-term layer corresponding to the empty subsequence. To solve this problem, our model deletes long-term layer nodes with empty inputs. However, if a node is simply deleted, the time interval is not considered correctly, and the recent time becomes the same as that long ago. However, the longer the time interval, the lesser the importance of sequencing information about what items were just before and the greater the need for a comprehensive review of past records. Therefore, the node in the long-term layer takes into account the time interval and uses the following $\hat{h}_t^l$ instead of the last node's hidden state, h_t^l

$$\hat{h}_t^l = \alpha^\tau h_t^l + (1-\alpha^\tau) h_{(mean,t)}^l$$

where τ is a time interval between two consecutive nodes, α is a constant, and $h_{(mean,t)}^l$ is the average of the hidden states in the long-term layer up to the previous time of t. This allows the hidden state of the long-term layer to gradually return to its mean over time. If $\tau \leq T_w$, then $\hat{h}_t^l = h_t^l$ because it is normal for no node to be deleted. That is,

$$\hat{h}_t^l = \begin{cases} \alpha^\tau h_t^l + (1-\alpha^\tau) h_{(mean,t)}^l & \tau > T_w \\ h_t^l & \tau \leq T_w \end{cases}.$$

V. EXPERIMENTAL RESULTS

A. DATASET

Our experiments used two real-world datasets: Movielens 1 million movie review dataset [21] and Steam game platform dataset [22]. We filter out users that have less than 5 events. Movielens dataset contains 6040 users, 3416 items, and 163.5 average review sequence length per user. Steam dataset contains 334730 users, 13047 items, and 12.26 average sequence length per user. We used only the implicit feedback from the dataset except for the rating, review contents, etc. That is, in the dataset we use, each user has a sequence consisting only of (item ID, timestamp) tuples. For each user, we used the most recent item for test, same as [23]. We used the second most recent item for verification and the rest for training.

B. BASELINES

We used common baseline methods, which were generally used in the recommendation system and compared them with our model. As described in Figure 6, the baseline methods were POP, BPR, FMC, FPMC, GRU4Rec and GRU4Rec+. Pop and BPR are methods of not considering sequences, FMC and FPMC are methods of considering the last visited item, and GRU4Rec(+) are RNN-based models.

- **Pop:** This model depends on how often each item was used previously. It only counts the total number of previous item uses.
- **BPR [3]:** This method applies Bayesian Personalized Ranking to matrix factorization.
- **FMC:** This method uses the last used item and a first-order marcov chain to recommend the next item.
- **FPMC [23]:** This method uses a combination of matrix factorization and markov chain. Therefore, unlike FMC, FPMC can reference not only the last item used but also the further past item.
- **GRU4Rec [4]:** This Session-based RNN method applies RNN to e-commerce to predict the items that the user will click on next through three layers (i.e. Embedding-GRU-Feedforward), based on items that the user has clicked on in the past; it uses a loss function based on item ranking. We use this for the movie dataset.
- **GRU4Rec+ [5]:** This model improves GRU4Rec by changing a loss function and a sampling strategy.

C. RESULTS

Here we use Hit@k as the measurement for comparison because Recall is well adapted to top-N recommendation system evaluation. Each method presents k candidates that

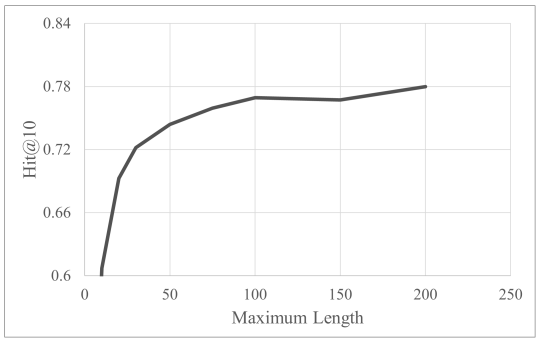


FIGURE 7. Hit@10 for movielens1M dataset with various maximum sequence lengths.

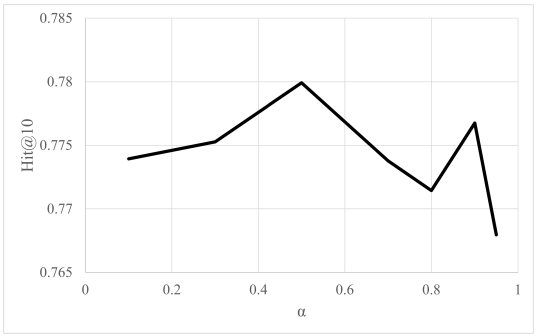


FIGURE 8. Hit@10 for movielens1M dataset with various α parameters.

are likely to be used next and we check the accuracy of whether the next item actually used was in the candidate group. Because this study predicts which item will be used at the next time, the correct answer is only one item. When predicting k candidate items, $Hit = \frac{\#correct}{\#user}$. To reduce the amount of calculation, we compared ground truth with 100 negative items that were randomly sampled instead of comparing it with all items [14].

Figure 6 shows the results of comparing the baseline models and the models presented in this study. We observe that the experimental results of our proposed model, Hi-RNN that consider the long-term user histories are significantly higher than those of other models. The hit@10 of the existing methods did not exceed 76%, so that of the model of this study was 78%. It is more than 10% different from Pop and BPR, which do not use any temporal property. GRU4Rec and GRU4Rec+ which consider the characteristics of sequence data by using RNN performs better than Pop and BPR, but the performance of the Hi-RNN model which consider time interval is better. The performance difference from Pop and BPR was not significant on the Steam dataset compared to the Movielens dataset. We guess that since the average sequence length is longer in the Movielens dataset than in the Steam dataset, the importance of the temporal property increases as the sequence becomes longer, resulting in this difference.

We conducted an additional experiment to further examine the difference in the importance of the temporal property according to the sequence length. Figure 7 is the result of a comparative experiment with the variable maximum sequence lengths on the Movielens dataset. This experiments show a significant decrease in hit rate to 0.6 when Hi-RNN looks only the last 10 records. This hit rate is more similar to BPR, which does not consider the sequence. Its performance is lower than that of FMC, which only looks at its last visit. However, as we saw above, when long periods of data were used together, hi-RNN's performance was the best. This shows that considering the temporal property is more meaningful in the long-term sequence. In addition, if we examine the last 200 items, we obtain a very slightly higher hit rate, of 78%, than 100 items. We guess that because the data that are too old are not properly reflected in the user's current characteristics and the current item issues.

Experiments on various α values as Figure 8 show the importance of time interval. The closer the α is to 1, the greater the reflection of the data just before, regardless of time interval. Conversely, if close to zero, the longer the time interval, the importance of the input just before becomes more like that just one of the past inputs. Experiments show the most improved performance when α is 0.5. This result means that the time interval should not be ignored.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a novel recommendation system that considers the order and time interval for the recommendation system. We construct the hierarchical model based on RNN that is effective in a long-term time series. We show that the model suggested in this study has high performance compared with several baseline models from experiments. The experiments indicates that it is important to use temporal properties in long-term time series datasets.

As for future work, we believe that the use of additional information together will further improve the model. For example, a user relationship graph-based information between users called ''trust'' can be used together. Previously, there have been many studies that have used trust together make more personalized and accurate recommendations [24]–[27]. However, these studies did not consider temporal properties such as sequential information and temporal interval. We expect that advancing the short-term layer of the Hi-RNN model, which currently use the vanilla RNN structure, will allow the model to consider not only the information of one user but also the information of both that user and the highly trust users together. In addition, we believe that other information such as review content and ratings can also be further considered to enhance the model.

REFERENCES

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, ''Item-based collaborative filtering recommendation algorithms,'' in *Proc. 10th Int. Conf. World Wide Web (WWW)*, 2001, pp. 285–295.
[2] Y. Hu, Y. Koren, and C. Volinsky, ''Collaborative filtering for implicit feedback datasets,'' in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 263–272.
[3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, ''BPR: Bayesian personalized ranking from implicit feedback,'' in *Proc. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[4] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, *arXiv:1511.06939*. [Online]. Available: https://arxiv.org/abs/1511.06939
[5] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-K gains for session-based recommendations," 2017, *arXiv:1706.03847*. [Online]. Available: https://arxiv.org/abs/1706.03847
[6] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proc. AAAI*, 2002, p. 1–6.
[7] T. Zhao, J. McAuley, and I. King, "Leveraging social connections to improve personalized ranking for collaborative filtering," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, Nov. 2014, pp. 261–279.
[8] L. Lerche and D. Jannach, "Using graded implicit feedback for Bayesian personalized ranking," in *Proc. 8th ACM Conf. Recommender Syst. (RecSys)*, 2014, pp. 353–356.
[9] G. Dahl, M. Ranzato, A. Mohamed, and G. Hinton, "Phone recognition with the mean-covariance restricted Boltzmann machine," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2010, pp. 1–9.
[10] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 536–543.
[11] F. Strub and J. Mary, "Collaborative filtering with stacked denoising autoencoders and sparse inputs," in *Proc. Workshop Mach. Learn. eCommerce (NIPS)*, Montreal, QC, Canada, 2015, pp. 150–158.
[12] H. Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1235–1244.
[13] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, Feb. 2016, pp. 153–162.
[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, Eds. Perth, NSW, Australia: ACM Press, Apr. 2017, pp. 173–182.
[15] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. INTERSPEECH*, 2010, pp. 1045–1048.
[16] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.
[17] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in NetEase," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 1218–1229.
[18] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, S. C. Kremer and J. F. Kolen, Eds. Piscataway, NJ, USA: IEEE Press, 2001.
[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *IEEE Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[20] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*. [Online]. Available: http://arxiv.org/abs/1409.1259
[21] F. Maxwell Harper and A. Joseph Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, Dec. 2015.
[22] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 197–206.
[23] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web (WWW)*, New York, NY, USA, 2010, pp. 811–820.
[24] D. Rafailidis, "Modeling trust and distrust information in recommender systems via joint matrix factorization with signed graphs," in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, New York, NY, USA, Apr. 2016, pp. 1060–1065.
[25] W. Li, X. Zhou, S. Shimizu, M. Xin, J. Jiang, H. Gao, and Q. Jin, "Personalization recommendation algorithm based on trust correlation degree and matrix factorization," *IEEE Access*, vol. 7, pp. 45451–45459, 2019.
[26] P. D. Meo, "Trust prediction via matrix factorisation," *ACM Trans. Internet Technol.*, vol. 19, no. 4, p. 44, Sep. 2019.
[27] H. Parvin, P. Moradi, S. Esmaeili, and N. N. Qader, "A scalable and robust trust-based nonnegative matrix factorization recommender using the alternating direction method," *Knowl.-Based Syst.*, vol. 166, pp. 92–107, Feb. 2019.

BYEONGJIN CHOE received the B.S. degree in electrical engineering from KAIST, Daejeon, South Korea, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea. His research interests include recommendation systems, sequential prediction, and machine learning and their applications to human behavior.

TAEGWAN KANG received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include natural language processing, and machine learning and applications.

KYOMIN JUNG (Member, IEEE) received the B.S. degree in mathematics from Seoul National University, Seoul, South Korea, in 2003, and the Ph.D. degree in mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2009. From 2009 to 2013, he was an Assistant Professor with the Department of Computer Science, KAIST. Since 2016, he has been an Assistant Professor and an Associate Professor with the Department of Electrical and Computer Engineering, Seoul National University (SNU). He is also an Adjunct Professor with the Department of Mathematical Sciences, SNU. His research interests include natural language processing, deep learning and applications, data analysis, and Web services.