Collaborative Learning Using LSTM-RNN for Personalized Recommendation

Benjamin A. Kwapong(✉), Richard Anarfi, and Kenneth K. Fletcher

University of Massachusetts Boston, Boston, MA 02125, USA
{benjamin.kwapong001,richard.anarfi001,kenneth.fletcher}@umb.edu

Abstract. Today, the ability to track users' sequence of online activities, makes identifying their evolving preferences for recommendation practicable. However, despite the myriad of available online activity information, most existing time-based recommender systems either focus on predicting some user rating, or rely on information from similar users. These systems, therefore, disregard the temporal and contextual aspects of users preferences, revealed in the rich and useful historical sequential information, which can potentially increase recommendation accuracy. In this work, we consider such rich, user online activity sequence, as a complex dependency of each user's consumption sequence, and combine the concept of collaborative filtering with long short-term memory recurrent neural network (LSTM-RNN), to make personalized recommendations. Specifically, we use encoder-decoder LSTM-RNN, to make sequence-to-sequence recommendations. Our proposed model builds on the strength of collaborative filtering while preserving individual user preferences for personalized recommendation. We conduct experiments using Movielens (https://grouplens.org/datasets/movielens) dataset to evaluate our proposed model and empirically demonstrate that it improves recommendation accuracy when compared to state-of-the-art recommender systems.

Keywords: Recommender systems · Deep learning · Neural networks · Recurrent neural networks · Long short-term memory RNN · Sequence-to-sequence recommendations

1 Introduction

The comfort, simplicity and extensive reach of the internet has altered the traditional approach to marketing and commerce leading to a dominant new brand, e-commerce and marketing. In this new form of service provision and commerce approach, users become overwhelmed by abundance and variety of products and services, resulting in the challenge of choice making. To ease this challenge the use of recommender systems (RS) has recently become a subject of interest.

RS are one of the most successful applications of data mining and machine learning technology in practice. They are typically based on the matrix completion problem formulation, where for each user-item-pair only one interaction (e.g., a rating) is considered [1]. Collaborative filtering (CF) is one of such

Q. Wang et al. (Eds.): SCC 2020, LNCS 12409, pp. 35–49, 2020.
https://doi.org/10.1007/978-3-030-59592-0_3

widely used and more effective service recommendation techniques. It bases its recommendations on the ratings or behavior of other users in the system [2,3]. Traditional memory- and model-based CF recommendation methods, although useful, are far from perfect, due to their disregard of time. Thus, they assume consumption events to be independent from each other, which precludes such methods from taking advantage of the temporal dynamics that naturally exist in user behavior, for personalized recommendation. This makes them unsuitable to capture the temporal aspects of recommendations, such as user evolving preferences or taste or context-dependent interests [4]. This is because, there are many application scenarios where considering short-term user interests and longer-term sequential patterns can be central to the success of a recommender system [1]. For instance, to predict the next best item from a user sequential events, sequential logs can also be used to derive longer-term behavior patterns, to detect interest drifts of individual users over time, identify short-term popularity trends in the community that can be exploited by recommendation algorithms, or to reason about the best point in time to remind users of certain items they have seen or purchased before [1].

Modern recurrent neural networks (RNN), such as the long short-term memory (LSTM), have proven very capable for sequence prediction problems and are well-suited to capture the evolution of users taste [5]. As a result, Devooght and Bersini [4] showed that CF can be viewed as a sequence prediction and demonstrated that by applying LSTM-RNN to CF recommendations. Their work however, does not consider user sequential event information and so fails to personalize recommendations. Similar to our proposed method is the works proposed by Ko et al. [6] and Donkers et al. [7]. In their work, Ko et al. [6] proposed a collaborative RNN for dynamic recommender systems. They studied sequential form of user event data and, by using ideas from CF, proposed a collaborative sequence model based on RNN. Also, Donkers et al. [7] proposed a sequential user-based RNN recommendation method. They showed, in their work, how individual users can be represented in addition to sequences of consumed items in a Gated Recurrent Unit (GRU), to effectively produce personalized next item recommendations.

These works, however, have some limitations. First, they are not powerful enough to represent and capture the complex dependencies that may exist within user event sequences, especially, when the sequences are very long and might be of variable lengths. Second, they fail to generate a distributed representation (embedding) of the input sequence, which reduces the task performance of their proposed models. Third, they base their recommendation on predicting either the next item in the sequence or a fixed number of items in a sequence and that makes them impractical, especially for applications where recommending items in variable sequence length sequence is expected.

To address the above limitations, this paper employs Encoder-Decoder LSTM-RNN, which is suitable for processing user sequence event data, for sequence-to-sequence recommendations. We build a flexible model to represent complex dependencies within long sequences, by building a stronger correlation

between user consumption sequence. We achieve this by modeling each user consumption instance as a dependency on all previous consumptions for personalize recommendations. The summary of our contributions are as follows:

1. We build a strong correlation between user consumption sequence by modeling each user's consumption instance as a dependency on all previous consumptions. This allows us the ability to represent complex dependencies within long sequences and explore the extra details and information embedded in sequential events in order to preserve user preferences.
2. We employ encoder-decoder LSTM-RNN because of the length of consumption preferences of users. LSTM-RNNs work better on long-term dependencies than traditional RNNs.
3. We build on the strength of collaborative filtering, by using other user's consumption preferences, while preserving individual user preferences and improves personalize recommendation accuracy. This bridges the gap created by most existing models, where recommendation is based on either modeling each user's individual preferences or describing all users by a single prototypical behavioral profile (global learning).
4. We perform experiments to evaluate our proposed model and compare it to baseline methods such as Bayesian Personalized Ranking Matrix Factorization [8] and Adaptive Hinge Pairwise Matrix Factorization [9].

The remainder of this paper is as follows. In Sect. 2 we discuss works related to sequence-to-sequence recommendations, then RNN-based CF recommendation methods and finally, personalized recommendations using RNN. Our proposed work is discussed in detail in Sect. 3. In Sect. 4, we present experiments to evaluate our proposed method and also discuss our results. Finally, the paper is concluded in Sect. 5.

2 Related Works

This section reviews several existing works in literature related to our proposed work. We also provide a distinction between our proposed method and existing related works.

2.1 Sequence-to-Sequence (seq2seq) Recommender Systems

Many real world problems can be modeled as sequence-to-sequence (seq2seq) problems [10–12]. Recurrent neural networks (RNNs) have proven to be an effective tool in seq2seq predictions. This has led to some very useful work in the area of seq2seq predictions using RNN techniques. Chu et al. [13] built a RNN for seq2seq prediction using GRU. The network treats a user's recent ratings or behaviors as an ordered sequence. Each of these user ratings or behaviors is modeled by the network's hidden layers. Furthermore, they integrate the GRU with back propagation neural network to increase the prediction accuracy. Hidasi et al. [14] proposed a session-based recommendation method by

modifying the basic GRU-RNN. The GRU-RNN modification was achieved by introducing session-parallel mini-batches based output sampling and ranking loss function. In their work, the network input is the actual state of the session while the output is the item of the next event in the session. For stability purposes, the input vector was normalized and this reinforced their memory effect.

In their work, Kuan et al. [15] proposed a Heterogeneous Attribute Recurrent Neural Networks (HA-RNN) model. HA-RNN combined sequence modeling and attribute embedding in item recommendation. Different from conventional RNNs, HA-RNN develops a hierarchical attribute combination mechanism to deal with variable lengths of attributes. The model uses attributes in the output layer and shares the parameters with the input layer to offer additional model regularization. It takes the union of identity and attributes as a sequence element and is able to capture the global sequential dependencies between items as well as between attributes.

Smirnova et al. [16] proposed a class of Contextual RNNs(CRNNs) for recommendation that can take into account the contextual information both in the input and output layers. Their method modifies the behavior of RNN by combining the context embedding with the item embedding and explicitly parameterizing the hidden unit transitions as a function of context information in the model dynamics.

Balakrishnan et al. [17] proposed a deep-playlist generation model, which uses LSTM-RNN to predict similarity between songs. Yang et al. [18] examined three state of the art deep neural network approaches: LSTM, Encoder-Decoder and Memory network in sequence prediction field to handle the software sequence learning and prediction task. Then, modified approaches based on these state of the art models were proposed to deal with additional information in sequence. These approaches focused on adding information to enrich embedding input of LSTM-RNN, adding a classifier to encoder-decoder neural network as an assisting model and processing data to be structured for memory unit in memory network.

2.2 Collaborative Filtering-Based Recommendations Using RNN

Often, when given a number of users with a record of their history, the next specific user consumption can be predicted in one of two ways; observing that user's history in isolation or finding similar users with a close consumption pattern. We review some related work focused on the latter. Devooght et al. [4] explored the use of RNN for the collaborative filtering problem. Using RNNs, they reframed collaborative filtering as a sequence prediction problem, leading to richer models and taking the evolution of users' taste into account. Their experiments showed that the LSTM-RNNs produce very good results on the Movielens and Netflix datasets, and is especially good in terms of short term prediction and item coverage as compared to standard nearest neighbors and matrix factorization methods. Their conclusions however, was based on the vanilla LSTM-RNN, the basic form of LSTM-RNN.

Similarly, leveraging user online activity sequences, Ko et al. [6] proposed a flexible and expressive collaborative sequence model based on RNNs. The model is designed to capture a user's contextual state as a personalized hidden vector by summarizing cues from a data-driven, thus variable, number of past time steps, and representing items by a real-valued embedding. They found that, by exploiting the inherent structure in the data, their formulation led to an efficient and practical method.

Another example of collaborative filtering-based sequence modeling can be seen in the work of Bansal et al. [19]. In their paper, they presented a method leveraging deep RNNs to encode a text sequence into a latent vector. GRUs were trained end-to-end to carry out the collaborative filtering task. In their application case study of scientific paper recommendation, the GRU training yielded models with significantly higher accuracy. Performance was further improved by multi-task learning, where the text encoder network is trained for a combination of content recommendation and item meta-data prediction.

2.3 Personalized Recommendation Using RNN

In the area of personalized recommendation Wu et al. [20], in their paper, outlined how they built a deep RNN (DRNN) to address the problem of collaborative filtering's failure to exploit current viewing history of the user which leads to an inability to provide a real-time customized recommendation. Their network tracks how users browse the website using multiple hidden layers. Each hidden layer models how the combinations of web pages are accessed and in what order. They developed an optimizer to automatically tune the parameters of their neural network to achieve a better performance. Their results on real world dataset showed that the DRNN approach outperforms previous collaborative filtering approaches significantly.

Donkers et al. [7] proposed how individual users can be represented in addition to sequences of consumed items in a new type of GRU, to effectively produce personalized next item recommendations. First, they used GRU-RNN to model the temporal dynamics of consumption sequences. Then, through a gated architecture with additional input layers, they explicitly represented an individual user. Their user-based GRUs were uniquely designed and optimized for the purpose of generating personalized next item recommendations.

Quadrana et al. [21] addressed the challenge of personalizing session-based recommendation by proposing a model based Hierarchical RNN (HRNN). Their HRNN model builds extra features on top of the standard RNN. First, there is an additional GRU layer to model information across user sessions and to track the evolution of the user interests over time. Also incorporated is a user-parallel mini-batch mechanism for efficient training.

To make the most out of encoder-decoder LSTM-RNNs, our methods in this work stand out from all of the above related works especially in how we practically capture and model the consumption complexities of the users for personalized recommendation. We build a very unique strong coherence between the various user events in each unique user consumption sequence. This especially

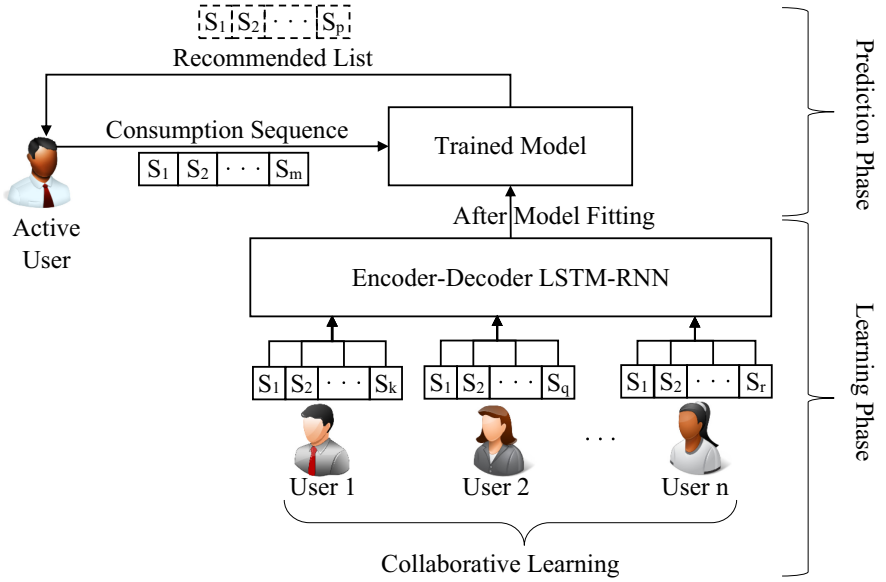

Fig. 1. Overview of the proposed collaborative learning model for personalized recommendation

helps different user patterns to be loosely coupled with each other, thereby better differentiating between unique user consumption patterns resulting in a more user aware model to improve personalized recommendation.

3 LSTM-RNN Based Collaborative Learning Model

In this section, we progressively give a detailed description of our proposed collaborative learning model for personalized recommendation. Figure 1 gives an overview of our proposed collaborative learning model for personalized recommendation. As shown in Fig. 1, our model comprises two main phases: learning phase and the prediction phase. In the learning phase, the timestamped user consumption sequence of a number of users are collated and fed into an encoder-decoder LSTM-RNN for model fitting to begin. After model fitting is complete, the trained model is now ready to make predictions based on user consumption sequences. With the fitted model, we feed in the consumption sequence of an active user to generate a list of items to be recommend. Section 3.2 discusses the technical details of our model.

Given a number of users and their item consumption history, how best can we make a personalized recommendation list of items to a specific user? As a case study, in this work, we use movies from the Movielens dataset as the item of consumption in question. Our quest, now, is to make a personalized movie recommendation for any user of our choice, based on the user's consumption history.

3.1 Problem Definition

Formally, let $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ be a set of users and $\mathcal{S} = \{s_1, s_2, ..., s_m\}$ be a set of services. We assume the number of users and services to be fixed. For each user $u \in \mathcal{U}$, we associate a consumption sequence $\mathcal{CS}(u) = [cs^u_{t_0}, cs^u_{t_1}, ..., cs^u_{t_k}]$, where each $cs^u_{t_k} \in \mathcal{S}$ and $t_0 \prec t_1 \prec ... \prec t_k$ denotes the time sequence of the service invocations. It must be noted that each service invocation in u's consumption sequence is an exclusive choice over $\mathcal{S}$. In addition, we focus on the consumption sequences to exploit the temporal order implicit in user consumption events.

Given a set of consumption sequences, $\mathcal{C} = \{\mathcal{CS}(u_1), \mathcal{CS}(u_2), ..., \mathcal{CS}(u_n)\}$ and a collaborative learning model $\mathfrak{L}$, we obtain a predictive model, $\mathfrak{P}$, after $\mathfrak{L}$ has learned on $\mathcal{C}$ over a period of time.

$$\mathfrak{L}(\mathcal{C}) :\longrightarrow \mathfrak{P} \tag{1}$$

Let a be an active user, such that $a \in \mathcal{U}$, with a consumption sequence, $\mathcal{CS}(a)$, we can predict the next p consumption sequence of a, using $\mathfrak{P}$.

3.2 Personalized Encoder-Decoder LSTM-RNN

The LSTM. Long Short-Term Memory (LSTM) networks are a special kind of Recurrent Neural Network (RNN), capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber [22] to address the vanishing gradient and exploding gradient issues in RNN, when the number of items in the sequence gets large (long term dependencies). Figure 2 shows a schematic diagram of a single LSTM block. An LSTM is composed of a cell, an input gate, an output gate and a forget gate. The major component is the cell state ("memory") which runs through the entire chain with occasional information updates from the input(add) and forget(remove) gates. An LSTM network computes a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$ by calculating the network unit activations using the following equations iteratively from $t = 1 to T$ [23]:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_t + b_i) \tag{2}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{4}$$

$$o_t = \sigma(w_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{5}$$

$$m_t = o_t \odot h(c_t) \tag{6}$$

$$y_t = \phi(W_{ym}m_t + b_y) \tag{7}$$

- f: forget gate's activation vector
- i: input gate's activation vector
- o: output gate's activation vector
- h: output vector of the LSTM unit

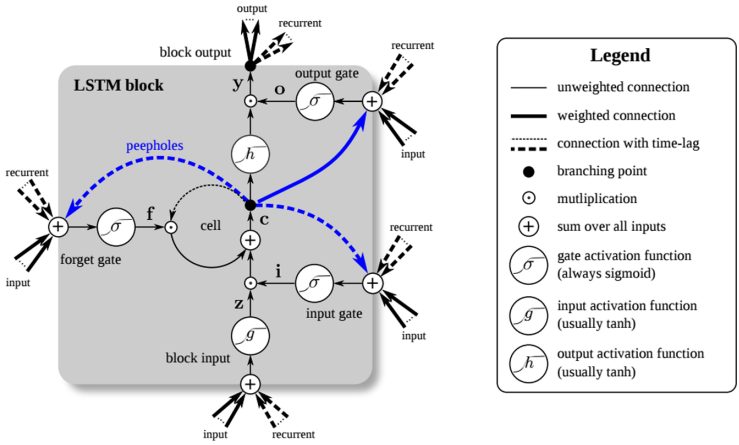

Fig. 2. Detailed schematic of an LSTM block as used in the hidden layers of a recurrent neural network [24]

- g: cell input activation function, generally tanh
- h: cell output activation functions, generally tanh
- c: cell activation vector
- W: weight matrices parameters
- b: bias vector parameters
- $\odot$: element-wise product of the vectors
- σ: the logistic sigmoid function
- ϕ: the network output activation function

The Encoder-Decoder. Figure 3 shows a simplified model of the encoder-decoder network architecture. An encoder is a network that takes the input and encodes it into an internal representation (feature/context vector), that holds the information and features, which best represents the input. The decoder is also a network that uses the vector from the encoder to generate an output sequence. In general, these networks only predict probabilities and the idea here is to first calculate the initial state of the input into a hidden state which is fed to the decoder to decode the information into the output sequence. A *softmax* takes the decoder's hidden state at time step t, and translates it into probability.

Our Model: The Personalized Encoder-Decoder LSTM-RNN. To prepare the data for training and subsequent testing, the input and expected output strings are tokenized into integers and the respective tokenizers are trained for our model. The encoded integers are then padded to the maximum input and output lengths respectively and the output sequence is one-hot encoded. We employ the encoder-decoder LSTM architecture to recommend a list of items for a user, based on his/her consumption preference. The choice of LSTM stems

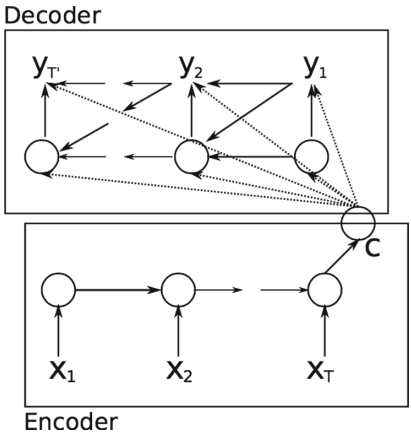

Fig. 3. A simplified illustration of an LSTM Encoder Decoder [25]

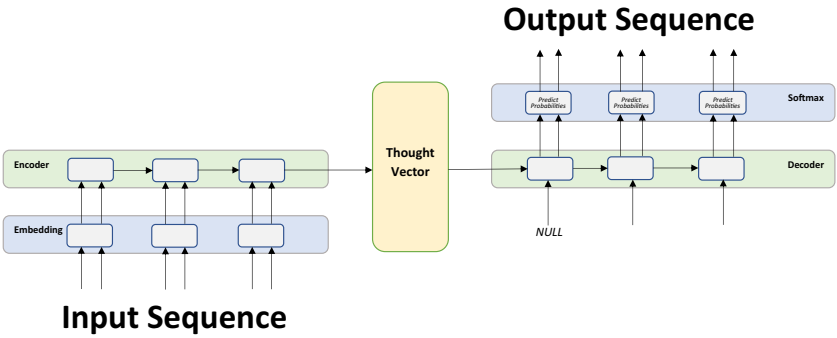

Fig. 4. Encoder-Decoder set-up for our experiments

from the fact that a comprehensive model and relationship will be learned from the user's consumption preference as well as the consumption preference of similar users thus introducing long-term dependencies [22]. The encoder-decoder architecture helps us to recommend a list of items to the user based on his/her consumption history and/or the consumption history of similar users. Figure 4 shows the basic set-up we used for our experiments. The encoder is basically a stack of LSTM cells. The thought vector is the final hidden state of the encoder. The actual outputs from the encoder are not passed to the decoder but rather the final hidden state. The decoder is also a stack of LSTM cells. The initial states of the decoder are set to the final states of the encoder. The model is trained using cross-entropy loss. At each step, the network produces a probability distribution over possible next tokens. The parameters for the model are carefully selected to provide the best training and subsequent prediction.

4 Experiments and Evaluation

We conducted several experiments to evaluate our proposed collaborative learning model for personalized recommendation. These experiments were done to ascertain the performance of our proposed model on seq2seq recommendations compared to other state of the art recommendation methods. Specifically, We considered a variation of our collaborative learning model where input to the encoder is reversed. The idea of reversed input comes from Sutskever et al. [11], where their tests proved that reversing input sequence presents some benefits to the model. In addition, we considered two variants of matrix factorization (MF) methods; Bayesian Personalized Ranking Matrix Factorization [8] and Adaptive Hinge Pairwise Matrix Factorization [9].

4.1 Dataset Description

We used the publicly available Movielens 10M[1] dataset. Movielens dataset is a benchmark dataset consisting of 10 million ratings and 100,000 tags from 72,000 users on 10,000 Movies. For each user, we obtained an ordered sequence of movie consumption using the timestamps in the dataset. Using the dataset, our goal is to predict the next n sequence of movies.

4.2 Baselines

Matrix Factorization (MF)-based methods have become classical technique for collaborative filtering as a result of its established success in recommendation systems [6]. In view of this, we find it plausible to compare our model to two MF based models.

- **Bayesian Personalized ranking Matrix Factorization (BPR-MF):** BPR-MF is a state of the art MF method for recommending Top-N items [4]. It is based on Bayesian Personalized Ranking loss function.
- **Adaptive Hinge Matrix Factorization (AHP-MF):** AHP-MF is based on the Adaptive hinge pairwise loss function (AHP). The AHP loss, in the SpotLight library, is an approximation for the Weighted Approximate-Rank Pairwise (WARP) loss scheme, proposed by Weston et al. [9]. According to Weston et al. [9], WARP loss yields better performance. For its competitiveness, we decided to consider AHP Matrix Factorization as one of our baselines.

[1] https://grouplens.org/datasets/movielens.

4.3 Metrics

The following metrics were chosen:

- **Recall:** Recall refers to sensitivity of the model. It captures the effectiveness of the model in terms of outputting relevant predictions. It can be computed as:

$$Recall(W_i) = \frac{tPositive(W_i)}{tPositive(W_i) + fNegative(W_i)}$$

- **Precision:** It assesses the predictive power of the algorithm [26].

$$Precision(W_i) = \frac{tPositive(W_i)}{tPositive(W_i) + fPositive(W_i)}$$

- **F-Measure:** This is defined on both recall and precision. It could be viewed as the weighted average of recall and precision. It rewards higher sensitivity [26].

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where $tPositive$, $fPositive$ and $fNegative$ are true positive, false positive and false negative respectively. Higher precision, recall and F-measure values indicates better performance.

4.4 Results and Discussions

We applied our method to the consumption preferences of 5000 users from the movielens dataset which translates into a combination of 625,000 different consumption preferences. In this section, we compare the results from our model with the results from the baselines described in the previous section.

Our main aim was to recommend a list of items (in this case, movies) to a user, based on what he has previously consumed and what other similar users have consumed. By similar users, we are referring to users who have similar consumption preferences and therefore are more likely to have similar future preferences.

All our LSTM models were fed with user consumption one after the other into the encoder and a sequence of outputs are obtained from the decoder. Details of the setup of our experiments are listed below:

1. 127,000 user consumption preferences with an encoder-decoder setup, both with a hidden state of 256 units. We trained the network over 20 epochs while updating the parameters using Adam optimization.
2. Input to encoder is reversed, 127,000 user consumption preferences with an encoder-decoder setup, both with a hidden state of 256 units. We trained the network over 20 epochs while updating the parameters using Adam optimization.
3. 625,000 user consumption preferences with an encoder-decoder setup, both with a hidden state of 256 units. We trained the network over 15 epochs while updating the parameters using Adam optimization.

Table 1. Training (Model) Parameters

Models	Learning_rate	Loss	Epochs	Batch	Emb_Dim	Optimizer
BPR-MF	0.05	bpr	10	256	32	ADAM
AHP-MF	0.01	Adaptive_Hinge	10	256	32	ADAM

4. Input to encoder is reversed, 625,000 user consumption preferences with an encoder-decoder setup, both with a hidden state of 256 units. We trained the network over 15 epochs while updating the parameters using Adam optimization. Table 1 shows the training parameters used for our baselines.

We evaluate the results of our experiments in terms of Recall@10, Precision@10 and F-Measure@10. Recall@k is equivalent to the hit-rate metric [21], and it measures the proportion of cases out of all test cases in which the relevant item is amongst the top-k items. This is an accurate model for certain practical scenarios where no recommendation is highlighted and their absolute order does not matter. Precision@k measures the fraction of correct recommendations in the top-k positions of each recommendation list. The training and validation accuracies from our experiments are captured in Figs. 5 and 6.

As expected, our experiments showed that increasing the number of users from 1000 to 5000 has a significant effect on the overall performance of the model. This shows that the collaboration from other users actually helps to improve the performance of our model by 27%. We also observed that reversing the input data gave an extra boost to the performance by a 14% margin. This shows that the idea of collaborative learning for personalized recommendation helps improve recommendation accuracy by a great deal.

From Table 2 BPR-MF had quite a good score (10%) in terms of precision as compared to AHP-MF. It was observed that the Adaptive Model (AHP-MF) produced low outputs, approximately 4.89% on recall, 7.26% precision and 5.84% F-measure.

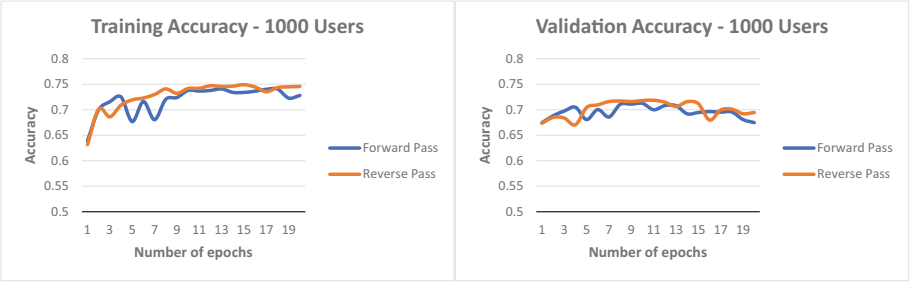

Fig. 5. A plot of training and validation accuracies for 1000 users

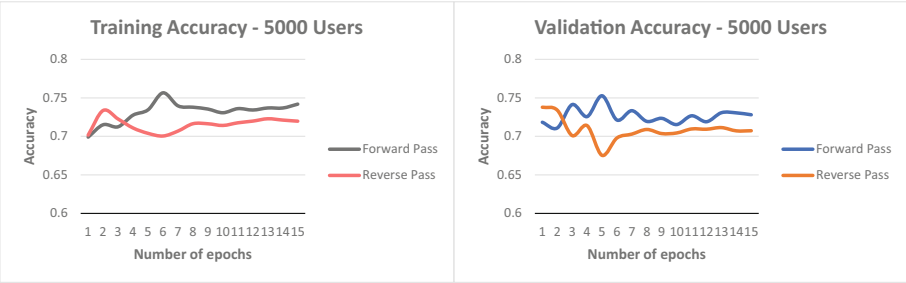

Fig. 6. A plot of training and validation accuracies for 5000 users

Table 2. Recall@k, Precision@k and F-Measure@k, (where k=10) on Movielens dataset

Model	Recall	Precision	F-Measure
BPR-MF	0.0679	0.1021	0.0816
AHP-MF	0.0489	0.0726	0.0584
Encoder-Decoder (1000 Users)	0.2068	0.2074	0.2071
Encoder-Decoder (1000 Users Reverse)	0.2454	0.2461	0.2457
Encoder-Decoder (5000 Users)	0.2723	0.2731	0.2727
Encoder-Decoder (5000 Users Reverse)	**0.2998**	**0.3006**	**0.3002**

5 Conclusions and Future Work

In this paper, we built a user consumption-sensitive Long Short-Term Memory recurrent neural network; specifically, an encoder-decoder model to tackle the real world problem of sequence to sequence prediction. Our model helped us to bridge the gap and benefit from the desirable attributes of both personalized recommendations based solely on a user's consumption history, and generic recommendation which is based on collaborative filtering. We have demonstrated that our method can significantly outperform popular baselines that are used for this task. We also noticed that the format in which the data is modeled has a very big impact on the performance of the network. In the near future, we will work on various data modeling formats and analyze which one is best suited for what purpose.

References

1. Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems. CoRR abs/1802.08452 (2018)
2. Fletcher, K.K.: A method for dealing with data sparsity and cold-start limitations in service recommendation using personalized preferences. In: 2017 IEEE International Conference on Cognitive Computing (ICCC), pp. 72–79, June 2017

3. Fletcher, K.K., Liu, X.F.: A collaborative filtering method for personalized preference-based service recommendation. In: Proceedings of the 2015 IEEE International Conference on Web Services, pp. 400–407, June 2015
4. Devooght, R., Bersini, H.: Collaborative filtering with recurrent neural networks. CoRR abs/1608.07400 (2016)
5. Kwapong, B.A., Anarfi, R., Fletcher, K.K.: Personalized service recommendation based on user dynamic preferences. In: Ferreira, J.E., Musaev, A., Zhang, L.J. (eds.) Services Computing - SCC 2019, pp. 77–91. Springer International Publishing, Cham (2019)
6. Ko, Y.J., Maystre, L., Grossglauser, M.: Collaborative recurrent neural networks for dynamic recommender systems. In: Journal of Machine Learning Research: Workshop and Conference Proceedings, vol. 63 (2016)
7. Donkers, T., Loepp, B., Ziegler, J.: Sequential user-based recurrent neural network recommendations. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 152–160. ACM (2017)
8. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
9. Weston, J., Bengio, S., Usunier, N.: WSABIE: scaling up to large vocabulary image annotation. IJCAI **11**, 2764–2770 (2011)
10. Park, S., Kim, B., Kang, C.M., Chung, C.C., Choi, J.W.: Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. arXiv preprint arXiv:1802.06338 (2018)
11. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
12. Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., Hinton, G.: Grammar as a foreign language. In: Advances in Neural Information Processing Systems, pp. 2773–2781 (2015)
13. Chu, Y., Huang, F., Wang, H., Li, G., Song, X.: Short-term recommendation with recurrent neural networks. In: 2017 IEEE International Conference on Mechatronics and Automation (ICMA), pp. 927–932. IEEE (2017)
14. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
15. Liu, K., Shi, X., Natarajan, P.: Sequential heterogeneous attribute embedding for item recommendation. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 773–780, November 2017
16. Smirnova, E., Vasile, F.: Contextual sequence modeling for recommendation with recurrent neural networks. arXiv preprint arXiv:1706.07684 (2017)
17. Balakrishnan, A., Dixit, K.: DeepPlaylist: using recurrent neural networks to predict song similarity (2016)
18. Yang, Q., He, Z., Ge, F., Zhang, Y.: Sequence-to-sequence prediction of personal computer software by recurrent neural network. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 934–940, May 2017
19. Bansal, T., Belanger, D., McCallum, A.: Ask the GRU: multi-task learning for deep text recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 107–114. ACM (2016)
20. Wu, S., Ren, W., Yu, C., Chen, G., Zhang, D., Zhu, J.: Personal recommendation using deep recurrent neural networks in NetEase. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 1218–1229.IEEE (2016)

21. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 130–137. ACM (2017)
22. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
23. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
24. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017)
25. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
26. Sokolova, M., Japkowicz, N., Szpakowicz, S.: Beyond accuracy, F-Score and ROC: a family of discriminant measures for performance evaluation. In: Sattar, A., Kang, B. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 1015–1021. Springer, Heidelberg (2006). https://doi.org/10.1007/11941439_114