Sequence-Aware Recommendation with Long-Term and Short-Term Attention Memory Networks

Daochang Chen
Tsinghua University
cdc16@mails.tsinghua.edu.cn

Rui Zhang, Jianzhong Qi
The University of Melbourne
{rui.zhang, jianzhong.qi}@unimelb.edu.au

Bo Yuan
Tsinghua University
yuanb@sz.tsinghua.edu.cn

Abstract—Next item recommendation is an important yet challenging task in real-world applications such as E-commerce. Since people often carry out a series of online shopping activities, in order to predict what a user may purchase next, it is essential to model the user's general taste as well as the sequential correlation between purchases. Existing models combine these two factors directly without considering the dynamic changes of a user's long-term and short-term preferences. Meanwhile, when a purchase session contains multiple items, not all of them have the same impact on the next item to purchase. To address these limitations, we propose a model that introduces hierarchical attention to dynamically balance between general taste (long-term preference) and sequential behavior (short-term preference). To weight individual items in the same session, we design a neural memory network with attention mechanism to learn the dynamic weights. Our model can adapt the embedding of each session as well as the embedding of long-term and short-term preferences. Extensive experiments on three real-world datasets show that our model significantly outperforms state-of-the-art methods based on commonly used evaluation metrics.

Keywords—Recommendation Systems; Sequential Correlation; Memory Networks; Attention Mechanism

I. Introduction

Recommender systems are becoming an important component of our daily life, especially in online services, including E-commerce, online news, and social media sites, where users' activities are influenced by their general tastes (long-term preference) as well as the sequential correlation (short-term preference) between purchases. The core task of a recommender system (RS) is to model and understand a user's preference to make a personalized recommendation[1].

For example, a user may produce a series of sessions, where each session contains a set of items purchased in a specific time (e.g, one day). The task is to recommend the most likely items to the user in the next session. We use context to summarize the factors that impact a user's next item to purchase. In particular, we consider both a user's general taste (long-term preference) that can be learned from his/her purchase history, and sequential behavior captured by the sequential dynamics (short-term preference).

Figure 1 gives an example showing a user's context changes and the recommendation results. Given two contexts A and B of the same user, the user's long-term preference is {Suits, Trousers} in both contexts, while the short-term preferences are different.

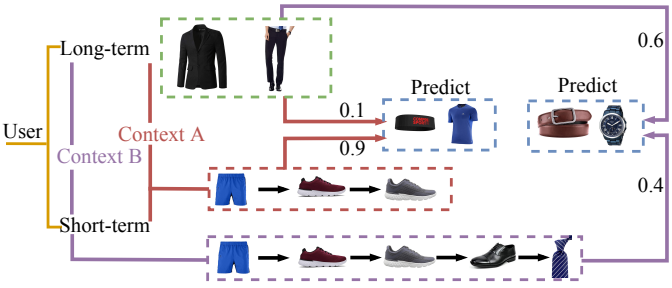

Fig. 1: Given a user $\boldsymbol{u}$, from his/her purchasing history, the long-term preference is focused on {Suits, Trousers}. For context A, {Sports T-shirts, Sweat Wristband} are recommended to the user, because his/her recent sequential behaviors mainly focus on sports-related items, and the long-term preference is irrelevant to sequential behaviors. For Context B, {Leather Belt, Dress Watch} are recommended to the user, because his/her recent purchases are mainly suit-related and both long-term and short-term preferences account for the recommendation.

In context A, the user's sequential behavior shows that his/her interest is on sports-related products. So, the short-term preference should be given higher weights. For example, if we assign weights 0.1 and 0.9 to long-term and short-term preferences, respectively, the system will recommend{Sports T-shorts, Sweat Wristband} to the user. In context B, the user's behavior shows that his/her interest changes from sports-related products to business suits. Since the user's long-term and short-term preferences are both related to business suits, we should make the weights relatively equal on long-term and short-term preferences. For example, by assigning weights 0.6 and 0.4 to long-term and short-term preferences, respectively, the system will recommend {Dress Watch, Leather Belt}.

To model the user's preference patterns, a variety of methods have been proposed. One class of methods recommends items based on a user's general taste (long-term preference), which assumes that users' preferences and items' profiles are relatively stable [2]. Another class

of methods only leverages the sequential dynamics, and adopt the Markov Chain (MC) [3] and recurrent neural network (RNN) based methods [4, 5], which ignores the user's general taste. Meanwhile, some methods jointly consider the user's general tastes and sequential transition behaviors [3, 6]. However, they only use a simple linear combination of the two factors, and the weighting between the two factors is fixed regardless of the user's current context. Moreover, existing methods do not assign individual weights to all items within a session [3], and thus cannot focus on the most relevant items. However, it is important to distinguish between relevant and irrelevant items, especially when a user has a long sequential session.

To address the above issues, we propose a novel long-term and short-term attention memory network (LSAMN). We exploit a two-level attention embedding memory network to bridge a user's general taste and the sequential behavior, and establish the connection among items with a session. To the best of our knowledge, this is the first attempt to introduce a mechanism to adapt the user's long-term and short-term attentions, while fully considering the weight of each item in the current session.

Our LSAMN model is different from the Sequential Hierarchical Attention Network (SHAN) model [7], which considers the long-term preference as a new item in the last session, and uses the last session to generate the hybrid representation of the user's preference. Compared with LSAMN, SHAN weakens the sequential influence on a user's preference. Meanwhile, only using a single vector to represent a user's preference inevitably limits the capability of the model.

The main contributions of this paper are summarized as follows:

- We propose a novel long-term and short-term attention memory network based on users' sequential session behaviors and general tastes for next item recommendation.
- Our proposed LSAMN model can adapt the long-term and short-term preferences dynamically, and capture the connection among items through the session level attention vector.
- We show that LSAMN significantly outperforms state-of-the-art recommendation approaches in terms of Recall and MRR on three real-world datasets.

II. Related Work

Next item recommendation has attracted extensive research interests. Most previous methods rely on Collaborative Filtering (CF) to model implicit feedbacks, among which Matrix Factorization (MF) is one of the most popular techniques [8]. Such models usually only consider a fixed user preference. However, real-world applications may contain a series of temporal sessions and the popularity of items may constantly change as the purchase behavior changes. Similarly, a user's preference for items can drift over time. Following this idea, sequential pattern recommender systems extract a series of association patterns according to the historical behaviors. Markov Chains have been used to model the sessions, and it has been shown that the sequential behavior mainly relies on the first-order session [3]. Since deep learning has achieved great success in session-based recommendation [5, 9], recent studies employed RNN-based methods to exploit the sequential behaviors of sessions [4, 5]. However, items in a session may not strictly follow a sequential order in many real-world scenarios.

To model a user's general taste and sequential dynamics jointly, Rendle et al. [3] linearly combined Matrix Factorization and Markov Chains to capture a user's preference pattern. Wang et al. [10] captured the hybrid representation of a user's preference by aggregation operations. However, these methods ignore both the intra-session items' attention and the preference level attention. By contrast, in document classification tasks, attention mechanism has shown great potential in modeling hierarchical patterns [11]. Bahdanau et al. [12] presented a technique for word alignment in machine translation using attention mechanism, which is similar to recommending relevant items. Our model is built on a two-level attention memory network, which can model a user's long-term and short-term preferences in two embedding spaces. As a result, our model can strategically combine a user's preferences by leveraging recurrent neural networks and the two-level attention mechanism, which fundamentally improves the model's capability.

III. Preliminary

We focus on modeling implicit (e.g., purchases, clicks) feedbacks. Let $U = \{u_1, u_2, \cdots, u_{|U|}\}$ denote a user set, where $|U|$ is the number of users in set U. Let $I = \{i_1, i_2, \cdots, i_{|I|}\}$ denote an item set and $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_{|I|}}\}$ be a set where each element $\boldsymbol{x_j}$ denotes the embedding of an item i_j. The purchase history of all users is denoted by $B = \{B^1, B^2, \cdots, B^{|U|}\}$ where $B^{u_j} = \{b_1^{u_j}, b_2^{u_j}, ...b_{|B^{u_j}|}^{u_j}\}$ represents the set of sessions of user u_j. Each session $b = \{i_1, i_2, \cdots, i_{|b|}\}$ consists of a subset of items. Given a sequence of sessions from user u, we aim to recommend top-K items for user u to purchase in the next session.

IV. Methodology

A. Overview of LSAMN

Figure 2 illustrates the framework of LSAMN, which consists of an embedding layer and a two-level attention memory component:

- An embedding network to embed the user and item IDs into two low-dimensional vector spaces.
- A session level attention memory network (SMN) to compute the current session embedding of each user.
- A preference level feed-forward neural memory network (PMN) to generate the long-term and short-term preference attention in the current context.

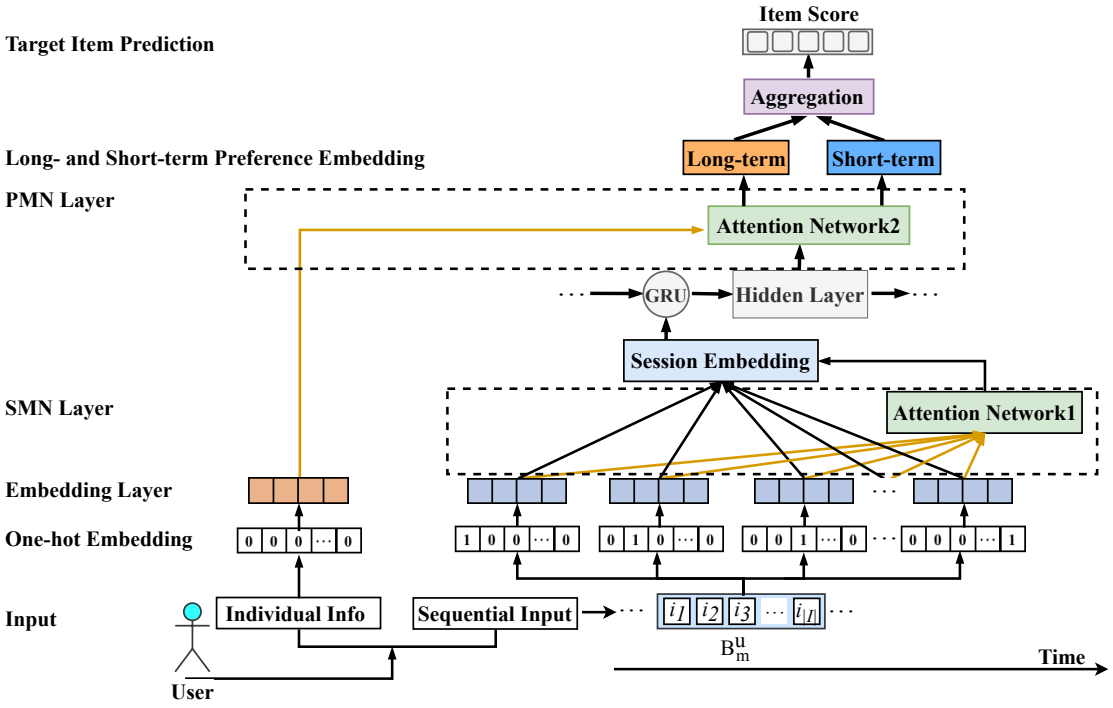

Fig. 2: The LSAMN model

B. LSAMN Components

1) The Embedding Layer: Given a user and a series of sequential sessions, we first map the sparse one-hot vectors of users and items to two continuous d-dimensional dense spaces, where $v \in \mathbb{R}^{d\times|U|}$ and $x \in \mathbb{R}^{d\times|I|}$ are the embeddings of users and items, respectively. v_u is used to represent the embedding of user u and x_i is used to represent the embedding of item i.

2) The SMN Layer: In sequence-based recommendation systems, each session can contain more than one items and may not follow strict sequential order, which may not be suitable for RNN-based methods. Therefore, learning a fixed-length embedding of each session is more reasonable. This sequential level attention memory network layer is used to form the session embedding, which can be computed as follows:

$$\boldsymbol{h_{xi}} = f(\boldsymbol{\Phi_0 x_i} + \boldsymbol{\Omega_0}) \tag{1}$$

$$\beta_i = \frac{exp(\boldsymbol{V^\beta h_{xi}})}{\sum_{k\in B_t^u} exp(\boldsymbol{V^\beta h_{xk}})} \tag{2}$$

where $\boldsymbol{V^\beta} \in \mathbb{R}^d$ denotes the weighting vector and β_i denotes the attention score of the i-th item in session B_t^u while $\boldsymbol{\Phi_0} \in \mathbb{R}^{\boldsymbol{d\times d}}$ and $\boldsymbol{\Omega_0} \in \mathbb{R}^{\boldsymbol{d}}$ are model parameters. Function $f(\cdot)$ is the activation function, and we use *sigmoid* to enhance the capacity of our model.

The session embedding can be calculated as follows:

$$\boldsymbol{b_t^u} = \sum_{i\in B_t^u}^{|B_t^u|} \beta_i \boldsymbol{x_i} \tag{3}$$

where $\boldsymbol{b_t^u}$ is the embedding of the current session in time t and β_i is the attention weight of the i-th item in session B_t^u, and $\boldsymbol{x_i} \in \mathbb{R}^d$ is the embedding of the i-th item in B_t^u.

3) The PMN Layer: The preference level attention is used to calculate the attention of long-term and short-term preferences. The parameters (α_l and α_s) are used to adaptively update long-term and short-term preferences. The embeddings of the long-term and short-term preferences can be computed as follows:

$$\boldsymbol{h_l^u} = f(\boldsymbol{\Phi_1 v_u} + \boldsymbol{\Omega_1}), \quad \hat{\boldsymbol{h}}_{\boldsymbol{l}}^{\boldsymbol{u}} = \alpha_l \cdot \boldsymbol{h_l^u} \tag{4}$$

$$\boldsymbol{h_s^u} = f(\boldsymbol{\Phi_2 h_t^u} + \boldsymbol{\Omega_2}), \quad \hat{\boldsymbol{h}}_{\boldsymbol{s}}^{\boldsymbol{u}} = \alpha_s \cdot \boldsymbol{h_s^u} \tag{5}$$

where $\boldsymbol{h_l^u}, \boldsymbol{h_s^u} \in \mathbb{R}^d$ denote the current output state and $\boldsymbol{h_t^u} \in \mathbb{R}^d$ is the hidden state, and $\boldsymbol{v_u}$ is the user's general preference. $\boldsymbol{\Phi_1}, \boldsymbol{\Phi_2} \in \mathbb{R}^{d\times d}$ are the weighting matrices and $\boldsymbol{\Omega_1}, \boldsymbol{\Omega_2} \in \mathbb{R}^d$ are the bias values. α_l and α_s represent the attention weights of the long-term and short-term preferences of user u, which are used to update a user's preference embedding in each context. $f(\cdot)$ is the activation function, which is *tanh* in our model.

The sequential memory cell is defined as:

$$\boldsymbol{h_t^u} = \psi(\boldsymbol{h_{t-1}^u}, \boldsymbol{b_t^u}, \boldsymbol{\Omega_t}) \tag{6}$$

where $\boldsymbol{b_t^u} \in \mathbb{R}^d$ is the session at time t, which can be computed by Equation 3. $\boldsymbol{h_{t-1}^u} \in \mathbb{R}^d$ is the hidden state of the last time and $\boldsymbol{\Omega_t}$ is a bias vector. $\psi(\cdot)$ represents the neural memory cell, which is the GRU cell in our model.

We use the attention network to adaptively update the embedding of a user's long-term and short-term preferences:

$$M_1 = \boldsymbol{V^\alpha} Q(\boldsymbol{\Phi_3 h_l^u} + \boldsymbol{\Phi_4 b_t^u} + \boldsymbol{\Omega_l}) \tag{7}$$

$$M_2 = \boldsymbol{V^\alpha} Q(\boldsymbol{\Phi_5 h_s^u} + \boldsymbol{\Phi_6 b_t^u} + \boldsymbol{\Omega_s}) \tag{8}$$

$$\alpha_l = \frac{exp(M_1)}{\sum_{j\in[1,2]} exp(M_j)} \tag{9}$$

$$\alpha_s = \frac{exp(M_2)}{\sum_{j\in[1,2]} exp(M_j)} \tag{10}$$

where $\boldsymbol{V^{\alpha}} \in \mathbb{R}^{1\times d}$ is a weighting vector shared by all contexts. $\boldsymbol{\Phi_3}, \boldsymbol{\Phi_4}, \boldsymbol{\Phi_5}, \boldsymbol{\Phi_6} \in \mathbb{R}^{d\times d}$ are weighting matrices. $\boldsymbol{\Omega_l}, \boldsymbol{\Omega_s} \in \mathbb{R}^d$ are the bias vectors. $Q(\cdot)$ is the activation function, which is $tanh(\cdot)$. In our model, α_l is the general taste preference in the current session while α_s is the sequential dynamic preference with $\alpha_l + \alpha_s = 1$.

4) Target Item Prediction: Finally, our score function is defined as follows:

$$\hat{\boldsymbol{y}}_{\boldsymbol{u,t}} = \boldsymbol{W_1}\hat{\boldsymbol{h}}_{\boldsymbol{l}}^{\boldsymbol{u}} + \boldsymbol{W_2}\hat{\boldsymbol{h}}_{\boldsymbol{s}}^{\boldsymbol{u}} + \boldsymbol{\Omega_t} \tag{11}$$

where $\boldsymbol{W_1}, \boldsymbol{W_2} \in \mathbb{R}^{|I|\times d}$ are the weighting matrices, connecting the embedding layer to the output layer. $\boldsymbol{\Omega_t} \in \mathbb{R}^N$ is a bias vector, and $\hat{\boldsymbol{y}}_{\boldsymbol{u,t}} \in \mathbb{R}^{|I|}$ denotes the score of all items for user u at current session t. $\hat{\boldsymbol{h}}_{\boldsymbol{l}}^{\boldsymbol{u}}$ and $\hat{\boldsymbol{h}}_{\boldsymbol{s}}^{\boldsymbol{u}}$ represent the user's long-term and short-term embeddings.

The output is the vector $\hat{\boldsymbol{Y}}_{\boldsymbol{u,t}} \in \mathbb{R}^{|I|}$, where each element $\hat{Y}_{u,t,i}$ is computed as Equation 12:

$$\hat{Y}_{u,t,i} = \frac{exp(\hat{y}_{u,t,i})}{Z(u,t)} \tag{12}$$

where $Z(u,t) = \sum_{i\in I} exp(\hat{y}_{u,t,i})$ is the normalization constant and $\hat{\boldsymbol{Y}}_{\boldsymbol{u,t}}$ denotes the probability distribution of the next purchase behaviors of user u with length $|I|$.

C. Parameter Learning

Given a user u and the associated session sequence, our goal is to recommend the next items that the user is likely to be interested, which requires the algorithm to rank the ground-truth item i higher than the none ground-truth item j. Here $i >_{u,t} j$ means that item i is ranked higher than item j for user u at time step t. The loss function of our method is indicated by Equation 13:

Algorithm 1: LSAMN Algorithm

Input: users U, items I, session set B, dimensions d, regularization λ, learning rate η
Output: model parameters Θ

```
while LSAMN not converaged do
    Shuffle the training datasets (u, {b_1^u, ..., b_{m-1}^u}, i, N_j)
    Initialize model parameters Θ
    for each user in U do
        for item i in the next basket b_m^u do
            Randomly select k negative sample j
            Compute the user's long-term preference h_l^u
              and short-term preference h_s^u according to
              Equations (1)∼(13)
            Compute one positive item score y_{u,m,i} and
              k negative item scores y_{u,m,j}
            Update model parameters Θ with stochastic
              gradient descent
        end
    end
end
Return model parameters Θ
```

$$L_\Theta = \sum_{u\in U}\sum_{B_t^u\in B^u}\sum_{i\in B_t^u} \log(\hat{Y}_{u,t,i}) - \lambda||\Theta||^2 \tag{13}$$

where λ is the regularization constant and Θ is the model parameters. The basic idea is to rank the observed item i higher than all non-observed items at time step t. Thus, the item's ranking can be defined as follows:

$$i >_{u,t} j \Leftrightarrow \hat{Y}_{u,t,i} > \hat{Y}_{u,t,j} \tag{14}$$

where $i \in B_t^u$ and $j \notin B_t^u$. Let D_{train} denote the training example and our model attempts to maximize the log probability. Since the item size is often very large, it is impractical to directly evaluate L_Θ and compute the log likelihood gradient. Thus, to train our model more efficiently, we use the negative sampling technique [13], and the objective function is defined as:

$$\begin{aligned} L_\Theta = \sum_{u\in U}\sum_{B_t^u\in B^u}\sum_{i\in B_t^u} & K\cdot\mathbb{E}_{j\sim Q}[\log\sigma(-\hat{y}_{u,t,j})]) \\ & + \log\sigma(\hat{y}_{u,t,i}) - \lambda||\Theta||^2 \end{aligned} \tag{15}$$

where $\sigma(\cdot) = 1/(1+e^{-x})$ and $Q(\cdot)$ denotes the negative sampling function, and K is the number of negative samples.

V. Experiments

A. Datasets

We conducted experiments on three Amazon[1] [14] datasets, including Sports and Outdoors (SO), Health and Personal Care (HPC), Clothing Shoes and Jewelry (CSJ). These three datasets contain user-product purchase sessions, and each session contains the items that a user purchased within a day.

Following previous studies [10], we cleaned up the datasets by removing long tail users and items: we removed all the items bought by less than 10 users and users that have bought less than 10 items; we also removed users with less than 3 sessions. The statistics of the cleaned datasets are shown in Table I.

B. Baseline Methods

We evaluated LSAMN against the following representative methods:

- POP: This model simply recommends the most popular item in training dataset.
- BPR-MF [15]: This model is a state-of-the-art matrix factorization method, which focuses on users' long-term preferences and ignores the sequential behaviors.
- FMC: This method is a Markov chain method, which predicts next items based on last session items.
- FPMC [3]: This method considers both users' long-term preferences and sequential dynamic behaviors for recommendation.
- HRM [10]: This model aggregates users' long-term preferences and their last sessions using aggregation operations such as max pooling and average pooling.

[1] http://jmcauley.ucsd.edu/data/amazon/

TABLE I: Statistics of datasets in our experiment

Dataset	#user $\|U\|$	#item $\|I\|$	#session	#sequence	avg.#session length	Sparsity
Health and Personal Care (HPC)	11.1k	33.0k	201944	136673	1.48	99.94%
Clothing Shoes and Jewelry (CSJ)	12.5k	61.2k	178510	108459	1.65	99.98%
Sports and Outdoors (SO)	10.2k	35.0k	162693	94043	1.75	99.95%

- DREAM [4]: This model is an RNN-based approach capable of capturing the global sequential feature of a user.
- SHAN [7]: This method employs two attention networks to represent a user's preference and then uses matrix factorization to compute each item's score as the MF method does.

C. Experimental Settings.

We used each user's last session B_t^u as the testing data, while all other sessions were used for training. Following previous studies [16], we generated a top-K list for each user u, where $K = \{10, 20\}$. The items in this list were recommended to the user for the next session. We compared items in the recommended list with those in the testing session of the user to evaluate the recommendation performance. We varied the following parameters in the experiments: the embedding layer dimensionality $d \in \{10, 20, 40, 60, 80, 100\}$, the regularization parameter $\lambda_\Theta \in \{0.1, 0.01, 0.001, 0.0001\}$, the learning rate $\lambda \in \{0.2, 0.02, 0.002, 0.0002\}$. We sampled five negative samples for each positive label, and set the batch size to 256 and the number of iterations to 300.

D. Evaluation Metrics

We employed two commonly used metrics REC@K and MRR for session-based recommendation evaluation [17]. A larger value of the metrics indicates better performance of the model:

- REC@K: This metric evaluates the recall of the top-K ranked items in the recommendation list over the testing sessions.
- MRR: This metric evaluates the mean reciprocal rank of the predictive positions of the ground-truth items on the testing sessions.

E. Results and Analysis

1) Overall Performance: The experiment results are summarized in Table II. The last row is the improvement of LSAMN relative to the best baseline. The results show that LSAMN outperformed all other methods according to REC@10, REC@20 and MRR on three real-world datasets. We see that sequence-based methods (e.g., FPMC and DREAM) outperformed non-sequential methods (e.g., BPR), which confirms the importance of considering sequential information. Meanwhile, FPMC outperformed FMC, demonstrating the effects of personalization. The advantage of LSAMN was particularly distinct on HPC and CSJ: it improved more than 28% (Recall@10) on CSJ, 33% (Recall@20) on HPC and around 14% (MRR) on HPC.

2) Influence of Attention Components: To evaluate the influence of attention, we conducted additional experiments to analyze the role of each component, as shown in Table III. N-LSAMN means no preference level or session level attention and S-LSAMN only considers session level attention, while P-LSAMN only considers preference level attention. It is clear that models with one-level attention outperformed the model without the attention network, indicating that the attention mechanism can better exploit the features of users and items. Furthermore, the model with the two-level attention memory network outperformed all other models, suggesting that the proposed two-level attention network is important for sequential recommendation tasks.

3) Influence of Hyper-Parameters: We investigated the influence of embedding dimension d on our LSAMN model over three Amazon datasets. Figure 3 shows its performance under REC@10, REC@20 and MRR for various dimensions without changing other hyper-parameters. The results show that LSAMN was relatively sensitive to the embedding dimension on HPC, while being reasonably stable on CSJ and SO. For all datasets, higher embedding dimensions did not necessarily result in better performance and LSAMN outperformed the competitive baselines by using relatively small dimensions.

VI. Conclusion

We proposed a novel neural attention memory network named LSAMN to bridge users' long-term preferences and sequential dynamics for personalized next item recommendation. LSAMN can effectively capture a user's long-term preference as well as his/her short-term preference in the current context, reducing the influence of irrelevant items and focusing on relevant items in the current session. Experiment results showed that LSAMN outperformed state-of-the-art models according to three popular metrics (Recall@10, Recall@20, MRR) on three real-world datasets. Compared with the best results of other state-of-the-art methods, LSAMN achieved up to 28%, 33% and 14% relative improvement based on the three metrics, respectively.

References

[1] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," in TOIS, 2004, pp. 143–177.

[2] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in ICDM, 2009, pp. 263–272.

TABLE II: Performance Comparison on Three Amazon Datasets.

Datasets	Sports and Outdoors (SO)			Health and Personal Care (HPC)			Clothing Shoes and Jewelry (CSJ)		
Metric(%)	Rec@10	Rec@20	MRR	Rec@10	Rec@20	MRR	Rec@10	Rec@20	MRR
POP	1.0180	1.6613	0.7526	1.7781	2.3846	1.2149	0.5634	0.8328	0.2817
BPRMF	0.9569	1.6451	0.7438	1.6850	2.0577	1.2052	0.5136	0.7875	0.2658
FMC	0.3055	0.5537	0.2182	1.7781	2.2166	1.2076	0.5165	0.8432	0.2645
FPMC	1.0234	1.6450	0.7538	1.7856	2.3211	1.2102	0.5541	0.8656	0.2820
HRM	0.9312	1.5877	0.7439	1.7871	2.3242	1.2102	0.5557	0.8268	0.2824
DREAM	0.9627	1.6012	0.6681	1.9865	2.4674	1.5403	0.5621	0.8873	0.2833
SHAN	0.9648	1.6894	0.6776	1.8894	2.2952	1.2677	0.5530	0.9073	0.2834
LSAMN	1.0297	1.6976	0.7655	2.3660	3.2886	1.7553	0.7215	1.0037	0.2910
Improvement(%)	0.62	0.49	1.55	19.10	33.28	13.96	28.06	10.62	2.68

TABLE III: Impact of Attention Components

Datasets	Sports and Outdoors (SO)			Health and Personal Care (HPC)			Clothing Shoes and Jewelry (CSJ)		
Metric (%)	Rec@10	Rec@20	MRR	Rec@10	Rec@20	MRR	Rec@10	Rec@20	MRR
N-LSAMN	0.9636	1.6313	0.7537	1.8141	2.6690	1.5745	0.5309	0.8638	0.2846
S-LSAMN	0.9957	1.6334	0.7541	1.8162	2.6843	1.5753	0.5531	0.8651	0.2854
P-LSAMN	0.9962	1.6469	0.7605	1.8897	2.7509	1.5858	0.5530	0.8647	0.2851
LSAMN	1.0297	1.6976	0.7655	2.3660	3.2686	1.7553	0.7215	1.0037	0.2910
Improvement(%)	3.36	3.08	0.64	25.21	18.82	10.69	30.45	16.02	2.07

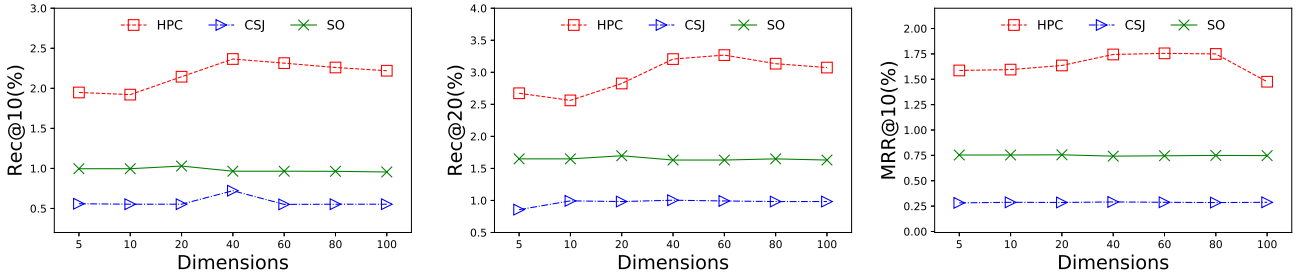

Fig. 3: The impact of embedding dimension d

[3] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in WWW, 2010, pp. 811–820.

[4] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in SIGIR, 2016, pp. 729–732.

[5] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in ICLR, 2016.

[6] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in ICDM, 2016, pp. 191–200.

[7] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention networks," in IJCAI, 2018, pp. 3926–3932.

[8] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, pp. 30–37, 2009.

[9] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," arXiv preprint arXiv:1707.07435, 2017.

[10] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning hierarchical representation model for nextbasket recommendation," in SIGIR, 2015, pp. 403–412.

[11] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in NAACL-HLT, 2017, pp. 1480–1489.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in ICLR, 2015.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in NIPS, 2013, pp. 3111–3119.

[14] R. He and J. Mcauley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in WWW, 2016, pp. 507–517.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in UAI, 2009, pp. 452–461.

[16] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu, "Diversifying personalized recommendation with user-session context," in IJCAI, 2017, pp. 1858–1864.

[17] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," TOIS, pp. 5–53, 2004.