Dynamic Memory based Attention Network for Sequential Recommendation

Qiaoyu Tan [1], Jianwei Zhang [2], Ninghao Liu [1], Xiao Huang [3]
Hongxia Yang [2], Jingren Zhou [2], Xia Hu [1]
[1] Texas A&M University, [2] Alibaba Group, [3] The Hong Kong Polytechnic University
{qytan,nhliu43,xiahu}@tamu.edu, xiaohuang@comp.polyu.edu.hk
{zhangjianwei.zjw, yang.yhx, jingren.zhou}@alibaba-inc.com

Abstract

Sequential recommendation has become increasingly essential in various online services. It aims to model the dynamic preferences of users from their historical interactions and predict their next items. The accumulated user behavior records on real systems could be very long. This rich data brings opportunities to track actual interests of users. Prior efforts mainly focus on making recommendations based on relatively recent behaviors. However, the overall sequential data may not be effectively utilized, as early interactions might affect users' current choices. Also, it has become intolerable to scan the entire behavior sequence when performing inference for each user, since real-world system requires short response time. To bridge the gap, we propose a novel long sequential recommendation model, called Dynamic Memory-based Attention Network (DMAN). It segments the overall long behavior sequence into a series of sub-sequences, then trains the model and maintains a set of memory blocks to preserve long-term interests of users. To improve memory fidelity, DMAN dynamically abstracts each user's long-term interest into its own memory blocks by minimizing an auxiliary reconstruction loss. Based on the dynamic memory, the user's short-term and long-term interests can be explicitly extracted and combined for efficient joint recommendation. Empirical results over four benchmark datasets demonstrate the superiority of our model in capturing long-term dependency over various state-of-the-art sequential models.

Introduction

Recommender systems have become an important tool in various online systems such as E-commerce, social media, and advertising systems to provide personalized services (Hidasi et al. 2015; Ying et al. 2018b). One core stage of live industrial systems is candidate selection and ranking (Covington, Adams, and Sargin 2016), which is responsible for retrieving a few hundred relevant items from a million or even billion scale corpus. Previously, researchers resort to collaborative filtering approaches (Sarwar et al. 2001) to solve it by assuming that like-minded users tend to exhibit similar preferences on items. Typical examples including models based on matrix factorization (Sarwar et al. 2001), factorization machines (Rendle 2010), and graph neural networks (Ying et al. 2018b; Wang et al. 2019b; Tan et al. 2020). However, these methods ignore the temporal dynamics of user behaviors.

To capture sequential dynamics for user modeling, various sequential recommendation methods have been proposed to make recommendations based on user's past behaviors (Hidasi et al. 2015; Tang and Wang 2018; Tan et al. 2021). They aim to predict the next item(s) that a user is likely to interact with, given her/his historical interactions. Recently, a myriad of attempts that build upon sequential neural networks, such as recurrent neural network (RNNs), convolutional neural networks (CNNs), and self-attention networks, have achieved promising results in various recommendation scenarios (Hidasi and Karatzoglou 2018; Yuan et al. 2019; Yan et al. 2019; Sun et al. 2019; Zhang et al. 2018). The basic paradigm is to encode a user's historical interactions into a vector using various sequential modules based on the behavior sequence, which is obtained by sorting her/his past behaviors in chronological order.

However, as many E-commerce and social media systems keep accumulating users' records, the behavior sequences have become extraordinarily long. For example, more than twenty thousand customers on the Alibaba e-commerce platform have interacted with over one thousand items from April to September 2018 (Ren et al. 2019). Despite the fertile information contained in these long behavior sequences, existing sequential recommendation algorithms would achieve sub-optimal performance in modeling long behavior sequences. The reason is that standard sequential architectures (e.g., RNNs, CNNs, attention networks) are insufficient to capture long-term dependencies confirmed in sequence learning (Graves, Wayne, and Danihelka 2014; Yu and Koltun 2015; Dai et al. 2019). Directly applying them to model long behavior sequences would result in significant performance degeneration. Thus, in this paper, we target at exploring sequential recommendation with extraordinary long user behavior sequences.

There are three major challenges in learning from long behavior sequences. 1) Given that the response time in real-world systems is limited, it has become expensive to scan over the entire behavior sequence at each prediction time (Zhu et al. 2018). Existing sequential recommender systems often require read the whole behavior sequence. A few recent long sequential recommendation methods explore splitting the whole input behavior sequence into short-term and long-term behavior sequences and then explicitly extracting a user's temporal and long-term preferences (Ying et al. 2018a; Lv et al. 2019). Despite their simplicity, they still suffer from high computation complexity since they need to scan over the whole behavior sequence during inference. 2) It is crucial to model the whole behavior sequence for a more accurate recommendation. A few attempts have been made to focus only on short-term actions (Li, Wang, and McAuley 2020; Hidasi and Karatzoglou 2018) and abandon previous user behaviors. Nevertheless, studies (Ren et al. 2019; Belletti, Chen, and Chi 2019) have demonstrated that user preferences may be influenced by her/his early interactions beyond the short-term behavior sequence. 3) It is hard to explicitly control the contributions of long-term or short-term interests for user modeling. Some studies resort to memory neural network (Graves, Wayne, and Danihelka 2014) to implicitly preserve the long-term intentions for efficient sequential modeling (Chen et al. 2018; Ren et al. 2019). But they may suffer from long-term knowledge forgetting (Sodhani, Chandar, and Bengio 2018), due to that the memory is optimized by predicting the next-item. Therefore, an advanced sequential model is needed to explicitly model both long-term and short-term preferences, as well as supporting efficient inference.

To address the limitations above, we propose a novel dynamic memory-based self-attention network, dubbed DMAN, to model long behavior sequence data. It offers standard self-attention networks to capture long-term dependencies for user modeling effectively. To improve model efficiency, DMAN truncates the whole user behavior sequence into several successive sub-sequences and optimizes the model sequence by sequence. Specifically, a recurrent attention network is derived to utilize the correlation between adjacent sequences for short-term interest modeling. Meanwhile, another attention network is introduced to measure dependencies beyond consecutive sequences for long-term interest modeling based on a dynamic memory, which preserves user behaviors before the adjacent sequences. Finally, the two aspect interests are adaptively integrated via a neural gating network for the joint recommendation. To enhance the memory fidelity, we further develop a dynamic memory network to effectively update the memory blocks sequence by sequence using an auxiliary reconstruction loss. To summarize, the main contributions of this paper are as follows:

- We propose a dynamic memory-based attention network DMAN for modeling long behavior sequences, which conducts an explicit and adaptive user modeling and supports efficient inference.
- We derive a dynamic memory network to dynamically abstract a user's long-term interests into an external memory sequence by sequence.
- Extensive experiments on several challenging benchmarks demonstrate our method's effectiveness in modeling long user behavior data.

The Proposed DMAN Model

In this section, we first introduce the problem formulation and then discuss the proposed framework in detail.

Table 1: Notations summary.

Notation	Description
u	a user
t	an item
$\mathbf{x}$	an interaction record
$\mathcal{U}$	the set of users
$\mathcal{V}$	the set of items
$\mathcal{S}_n$	the n-th behavior sequence
K	the number of candidate items
N	the number of sliced sequences
L	the number of self-attention layers
m	the number of memory slots
D	the number of embedding dimension
$\widetilde{\mathbf{H}}$	the short-term interest embedding
$\widehat{\mathbf{H}}$	the long-term interest embedding
$\mathbf{M}$	the memory embedding matrix
$\mathbf{V}$	the output user embedding

Notations and Problem Formulation

Assume $\mathcal{U}$ and $\mathcal{V}$ denote the sets of users and items, respectively. $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{|\mathcal{S}|}\}$ represents the behavior sequence in chronological order of a user. $\mathbf{x}_t \in \mathcal{V}$ records the t-th item interacted by the user. Given an observed behavior sequence $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t\}$, the sequential recommendation task is to predict the next items that the user might be interacted with. Notations are summarized in Table 1.

In our setting, due to the accumulated behavior sequence $\mathcal{S}$ is very long, we truncate it into a series of successive sub-sequences with fixed window size T, i.e., $\mathcal{S} = \{\mathcal{S}_n\}_{n=1}^{N}$, for the model to process efficiently. $\mathcal{S}_n = \{\mathbf{x}_{n,1}, \mathbf{x}_{n,2}, \dots, \mathbf{x}_{n,T}\}$ denotes the n-th sequence. Traditional sequential recommendation methods mainly rely on a few recent behaviors $\mathcal{S}_N$ for user modeling. Our paper focuses on leveraging the whole behavior sequence for a comprehensive recommendation. We first illustrate how to explicitly extract short-term and long-term user interests from historical behaviors, and then describe an adaptive way to combine them for joint recommendation. Finally, we introduce a novel dynamic memory network to preserve user's long-term interests for efficient inference effectively.

Recurrent Attention Network

This subsection introduces the proposed recurrent attention network for short-term interest modeling. Given an arbitrary behavior sequence $\mathcal{S}_n$ as input, an intuitive way to estimate a user's short-term preferences is only consider her/his behaviors within the sequence. However, the first few items in each sequence may lack necessary context for effective modeling, because previous sequences are not considered.

To address this limitation, we introduce the notion of recurrence in RNNs into self-attention network and build a sequence-level recurrent attention network, enabling information flow between adjacent sequences. In particular, we use the hidden state computed for last sequence as additional context for next sequence modeling. Formally, let $\mathcal{S}_{n-1}$ and $\mathcal{S}_n$ be two successive sequences, and $\widetilde{\mathbf{H}}_{n-1}^{l} \in \mathbb{R}^{T \times D}$ denote the l-th layer hidden state produced for sequence $\mathcal{S}_{n-1}$. We

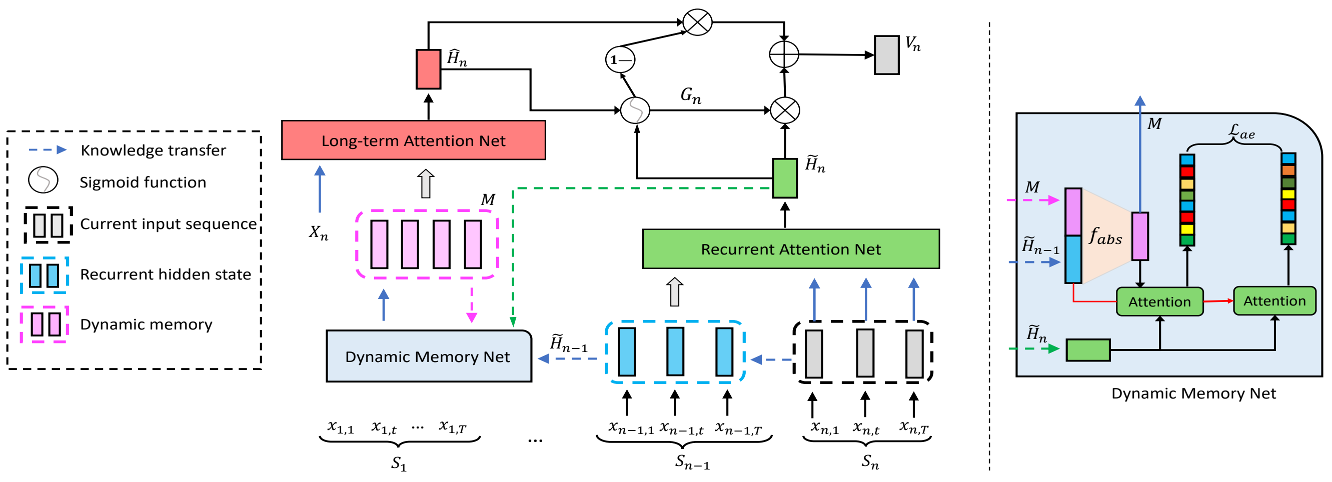


Figure 1: Illustration of DMAN for one layer. It takes a series of sequences as input and trains the model sequence by sequence. When processing the n-th sequence $\mathcal{S}_n$, the recurrent attention network is applied to extract short-term user interest by using the previous hidden state $\widetilde{\mathbf{H}}_{n-1}$ as context. Meanwhile, the long-term attention network is utilized to extract long-term interest based on the memory blocks $\mathbf{M}$. Next, the short-term and long-term interests are combined via a neural gating network for joint user modeling. Finally, the dynamic memory network updates the memory blocks via fusing the information in $\widetilde{\mathbf{H}}_{n-1}$, and the model continues to process the next sequence. The overall model is optimized by maximizing the likelihood of observed sequence, while the dynamic memory network is trained based on a local reconstruction loss $\mathcal{L}_{ae}$.

calculate the hidden state of sequence $\mathcal{S}_n$ as follows.

$$\begin{aligned}
\widetilde{\mathbf{H}}^l_n &= \text{Atten}^l_{\text{rec}}(\widetilde{\mathbf{Q}}^l_n, \widetilde{\mathbf{K}}^l_n, \widetilde{\mathbf{V}}^l_n) = \text{softmax}(\widetilde{\mathbf{Q}}^l_n (\widetilde{\mathbf{K}}^l_n)^\top)\widetilde{\mathbf{V}}^l_n,\\
\widetilde{\mathbf{Q}}^l_n &= \widetilde{\mathbf{H}}^{l-1}_n \widetilde{\mathbf{W}}^\top_Q, \quad \text{and} \quad \widetilde{\mathbf{K}}^l_n = \mathbf{H}^{l-1}_n \widetilde{\mathbf{W}}^\top_K,\\
\widetilde{\mathbf{V}}^l_n &= \mathbf{H}^{l-1}_n \widetilde{\mathbf{W}}^\top_V,\\
\mathbf{H}^{l-1}_n &= \widetilde{\mathbf{H}}^{l-1}_n \parallel \text{SG}(\widetilde{\mathbf{H}}^{l-1}_{n-1}),
\end{aligned} \tag{1}$$

where $\text{Atten}^l_{\text{rec}}(\cdot,\cdot,\cdot)$ is the l-th layer self-attention network, in which the query, key and value matrices are denoted by $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$, respectively. The input of the first layer is the sequence embedding matrix $\mathbf{X}_n = [\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,T}] \in \mathbb{R}^{T\times D}$. Intuitively, the attention layer calculates a weighted sum of embeddings, where the attention weight is computed between query i in $\mathcal{S}_n$ and value j obtained from previous sequences. The function $\text{SG}(\cdot)$ stands for stop-gradient from previous hidden state $\widetilde{\mathbf{H}}^{l-1}_{n-1}$, and $\parallel$ denotes concatenation. In our case, we use the extended context as key and value and adopt three linear transformations to improve the model flexibility, where $\{\widetilde{\mathbf{W}}_Q, \widetilde{\mathbf{W}}_K, \widetilde{\mathbf{W}}_V\} \in \mathbb{R}^{D\times D}$ denote the parameters. The extended context not only provides precious information for recovering the first few items, but also allows our model to capture the dependency across sequences. In practice, instead of computing the hidden states from scratch at each time point, we cache the hidden state of last sequence for reuse. Besides, masking strategy and positional embeddings are also included to avoid the future information leakage problem (Yuan et al. 2019) and capture sequential dynamics (Kang and McAuley 2018; Wang et al. 2019a). The final short-term interest embedding is defined as $\widetilde{\mathbf{H}}_n = \widetilde{\mathbf{H}}^L_n$.

Long-term Attention Network

In this subsection, we present another attention network for long-term interest modeling. With the recurrent connection mechanism defined in Eq. 1, our model can capture correlations between adjacent sequences for interest modeling. However, longer-range dependencies beyond successive sequences may still be ignored, since the recurrent connection mechanism is limited in capturing longer-range correlations (Sodhani, Chandar, and Bengio 2018; Sukhbaatar et al. 2015). Hence, additional architecture is needed to effectively capture long-term user preferences.

To this end, we maintain an external memory matrix $\mathbf{M} \in \mathbb{R}^{m\times D}$ to explicitly memorize a user's long-term preferences, where m is the number of memory slots. Each user is associated with a memory. Ideally, the memory complements with the short-term interest modeling, with the aim to capture dependencies beyond adjacent sequences. We leave how to effectively update the memory in later section and now focus on how to extract long-term interests from the memory. Specifically, let $\mathbf{M}^l \in \mathbb{R}^{m\times D}$ denote the l-th layer memory matrix, we estimate the long-term hidden state of sequence $\mathcal{S}_n$ using another self-attention network as

$$\begin{aligned}
\widehat{\mathbf{H}}^l_n &= \text{Atten}^l(\widehat{\mathbf{Q}}^l_n, \widehat{\mathbf{K}}^l_n, \widehat{\mathbf{V}}^l_n),\\
\widehat{\mathbf{Q}}^l_n, \widehat{\mathbf{K}}^l_n, \widehat{\mathbf{V}}^l_n &= \widehat{\mathbf{H}}^{l-1}_n \widehat{\mathbf{W}}^\top_Q, \mathbf{M}^{l-1}\widehat{\mathbf{W}}^\top_K, \mathbf{M}^{l-1}\widehat{\mathbf{W}}^\top_V.
\end{aligned} \tag{2}$$

Similarly, $\text{Atten}^l(\cdot,\cdot,\cdot)$ is a self-attention network. It takes the last layer hidden state $\widehat{\mathbf{H}}^{l-1}_n$ as query and uses the layer-wise memory matrix $\mathbf{M}^{l-1}$ as key (value). The input of the query is $\mathbf{X}_n$. By doing so, the output hidden state $\widehat{\mathbf{H}}^l_n \in \mathbb{R}^{T\times D}$ is a selective aggregation of m memory blocks, where the selection weight is query-based and varies across different queries. $\{\widehat{\mathbf{W}}_Q, \widehat{\mathbf{W}}_K, \widehat{\mathbf{W}}_V\}$ are trainable transformation matrices to improve model capacity. Since the memory is maintained to cache long-term user interests that beyond adjacent sequences, we refer to the above attention network as long-term interest modeling. The final long-term interest embedding for sequence $\mathcal{S}_k$ is denoted as $\widehat{\mathbf{H}}_n = \widehat{\mathbf{H}}_n^L$.

Neural Gating Network

After obtaining the short-term and long-term interest embeddings, the next aim is to combine them for comprehensive modeling. Considering that a user's future intention can be influenced by early behaviors, while short-term and long-term interests may contribute differently for next-item prediction over time (Ma et al. 2019), we apply a neural gating network to adaptively control the importance of the two interest embeddings.

$$\begin{aligned} \mathbf{V}_n &= \mathbf{G}_n \odot \widetilde{\mathbf{H}}_n + (1 - \mathbf{G}_n) \odot \widehat{\mathbf{H}}_n, \\ \mathbf{G}_n &= \sigma(\widetilde{\mathbf{H}}_n \mathbf{W}_{short} + \widehat{\mathbf{H}}_n \mathbf{W}_{long}), \end{aligned} \tag{3}$$

where $\mathbf{G}_n \in \mathbb{R}^{T\times D}$ is the gate matrix learned by a non-linear transformation based on short-term and long-term embeddings. $\sigma(\cdot)$ indicates the sigmoid activation function, $\odot$ denotes element-wise multiplication, and $\mathbf{W}_{short}, \mathbf{W}_{long} \in \mathbb{R}^{D\times D}$ are model parameters. The final user embedding $\mathbf{V}_n \in \mathbb{R}^{T\times D}$ is obtained by a feature-level weighted sum of two types of interest embeddings controlled by the gate.

Dynamic Memory Network

In this subsection, we describe how to effectively update the memory $\mathbf{M}$ to preserve long-term user preferences beyond adjacent sequences. One feasible solution is to maintain a fixed-size FIFO memory to cache long-term message. This strategy is sub-optimal for user modeling due to two reasons. First, the oldest memories will be discarded if the memory is full, whether it is important or not. This setting is reasonable in NLP task (Rae et al. 2019) as two words that too far away in the sentence are often not correlated. But it is not held in recommendation because behavior sequence is not strictly ordered (Yuan et al. 2019) and users often exhibit monthly or seasonal periodic behaviors. Second, the memory is redundant and not effectively utilized, since user interests in practice is often bounded in tens (Li et al. 2019).

To avoid these limitations, we propose to abstract a user's long-term interests from the past actively. Assume the model was processed sequence $\mathcal{S}_n$, then the memory is updated as

$$\mathbf{M}^l \leftarrow f_{abs}^l(\mathbf{M}^l, \widetilde{\mathbf{H}}_{n-1}^l), \tag{4}$$

where $f_{abs}^l : \mathbb{R}^{(m+T)\times D} \to \mathbb{R}^{m\times D}$ is the l-th layer abstraction function. It takes the old memory and the context state $\widetilde{\mathbf{H}}_{n-1}^l$ as input and updates memory $\mathbf{M}^l$ to represent user interests. In theory, f_{abs} requires to effectively preserve the primary interests in old memories and merges contextual information. Basically, f_{abs} can be trained with the next-item prediction task end-to-end. Nevertheless, memories that differ from the target item may be discarded. Therefore, we consider train the abstraction function with an auxiliary attention-based reconstruction loss as follows.

$$\begin{aligned} \mathcal{L}_{ae} &= \min \sum_{l=1}^{L} ||\text{attent}_{rec}^l(\widetilde{\mathbf{Q}}^l, \widetilde{\mathbf{K}}^l, \widetilde{\mathbf{V}}^l) - \text{attent}_{rec}^l(\widetilde{\mathbf{Q}}^l, \widehat{\mathbf{K}}^l, \widehat{\mathbf{V}}^l)||_F^2 \\ \widetilde{\mathbf{Q}}^l &= \widetilde{\mathbf{H}}_n^l,\ \widetilde{\mathbf{K}}^l = \widetilde{\mathbf{V}}^l = \mathbf{M}^l \parallel \widetilde{\mathbf{H}}_{n-1}^l,\ \widehat{\mathbf{K}}^l = \widehat{\mathbf{V}}^l = \mathbf{M}^l, \end{aligned} \tag{5}$$

where $\text{atten}_{rec}^l(\cdot,\cdot,\cdot)$ is the self-attention network defined in Eq. (1). We reuse the recurrent attention network but keep the parameters fixed and not trainable here. We employ the hidden state $\widetilde{\mathbf{H}}_n^l$ of $\mathcal{S}_n$ as query for two attention networks. The first attention outputs a new representation for the query via a weighted sum from the old and new memories, while the second from the abstracted memories. By minimizing the reconstruction loss, we expect the primary interests can be extracted by f_{abs} as much as possible. Note that we consider a lossy objective here because the information that is no longer attended to in $\mathcal{S}_n$ can be discarded in order to capture the shifting of user interests to some extent.

Implementation of abstraction function f_{abs} We parameterize f_{abs} with the dynamic routing method in CapsNet (Sabour, Frosst, and Hinton 2017) for its promising results in capturing user's diverse interests in recommendation (Li et al. 2019). Suppose we have two layers of capsules, we refer capsules from the first layer and the second layer as primary capsules and interest capsules, respectively. The goal of dynamic routing is to calculate the values of interest capsules given the primary capsules in an iterative fashion. In each iteration, given primary capsules vectors $\mathbf{x}_i$ (input vector), $i \in \{1, \ldots, T+m\}$ and interest capsules $\bar{\mathbf{x}}_j$ (output vector), $j \in \{1, \ldots, m\}$, the routing logit b_{ij} between primary capsule i and interest capsule j is computed by

$$b_{ij} = \bar{\mathbf{x}}_j^\top \mathbf{W}_{ij} \mathbf{x}_i, \tag{6}$$

where $\mathbf{W}_{ij}$ is a transformation matrix. Given routing logits, $\mathbf{s}_j$ is computed as weighted sum of all primary capsules

$$\mathbf{s}_j = \sum_{i=1}^{m+T} \alpha_{ij} \mathbf{W}_{ij} \mathbf{x}_i, \tag{7}$$

where $\alpha_{ij} = \exp(b_{ij}) / \sum_{j'=1}^{m+T} \exp(b_{ij'})$ is the connection weight between primary capsule i and interest capsule j. Finally, a non-linear "squashing" function (Sabour, Frosst, and Hinton 2017) is proposed to obtain the corresponding vectors of interest capsules as

$$\bar{\mathbf{x}}_j = \text{squash}(\mathbf{s}_j) = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}. \tag{8}$$

The routing process between Eq. (6) and Eq. (8) usually repeats three times to converge. When routing finishes, the output interest capsules of user u are then used as the memory, i.e., $\mathbf{M} = [\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m]$.

Model Optimization

As the data is derived from the user implicit feedback, we formulate the learning problem as a binary classification

Table 2: The dataset statistics.

Dataset	#Users	#Items	T	K
MovieLens	6,040	3,952	20	10
Taobao	987,994	4,162,024	20	10
JD.com	1,608,707	378,457	20	10
XLong	20,000	3,269,017	50	20

task. Given the training sample (u, t) in a sequence $\mathbf{S}_n$ with the user embedding vector $\mathbf{V}_{n,t}$ and target item embedding $\mathbf{x}_t$, we aim to minimize the following negative likelihood

$$\begin{aligned}\mathcal{L}_{like} &= -\sum_{u\in\mathcal{U}}\sum_{t\in\mathcal{S}_n}\log P(\mathbf{x}_{n,t}|\mathbf{x}_{n,1},\mathbf{x}_{n,2},\cdots,\mathbf{x}_{n,t-1}) \\ &= -\sum_{u\in\mathcal{U}}\sum_{t\in\mathcal{S}_n}\log\frac{\exp(\mathbf{x}_t^\top\mathbf{V}_{n,t})}{\sum_{j\in\mathcal{V}}\exp(\mathbf{x}_j^\top\mathbf{V}_{n,t}))}.\end{aligned} \tag{9}$$

The loss above is usually intractable in practice because the sum operation of the denominator is computationally prohibitive. Therefore, we adopt a negative sampling strategy to approximate the softmax function in experiments. When the data volume is large, we leverage Sampled Softmax technique (Covington, Adams, and Sargin 2016; Jean et al. 2014) to further accelerate the training. Note that Eqs. (9) and (6) are separately updated in order to preserve long-term interests better. Specifically, we first update Eq. (9) by feeding a new sequence and then updating the abstraction function's parameters by minimizing Eq. (6).

Experiments and Analysis

Datasets

We conduct experiments over four public benchmarks. Statistics of them are summarized in Table 2. MovieLens [1] collects users' rating scores for movies. JD.com (Lv et al. 2019) is a collection of user browsing logs over e-commerce products collected from JD.com. Taobao (Zhu et al. 2018) and XLong (Ren et al. 2019) are datasets of user behaviors from the commercial platform of Taobao. The behavior sequence in XLong is significantly longer than other three datasets, thus making it difficult to model.

Baselines

To evaluate the performance of DMAN, we include three groups of baseline methods. First, traditional sequential methods. To evaluate the effectiveness of our model in dealing with long behavior sequence, three state-of-the-art recommendation algorithms for sequences with a normal length have been employed, including **GRU4Rec** (Tang and Wang 2018), **Caser** (Kang and McAuley 2018) and **SASRec** (He and Chua 2017). Second, long sequential methods. To evaluate the effectiveness of our model in extracting long-term user interests with dynamic memory, we include **SDM** (Lv et al. 2019) and **SHAN** (Ying et al. 2018a), which are tailored for modeling long behavior sequences. To evaluate the effectiveness of our model in explicitly capturing user's short-term and long-term interests, we also set **HPMN** (Ren et al. 2019) as a baseline. It is based on the memory network. Thrid, DMAN variants. To analyze the contribution of each component of DMAN, we consider three variants. DMAN-XL discards the long-term attention network to verify the effectiveness of capturing long-term interests. DMAN-FIFO adopts a FIFO strategy to validate the usefulness of the abstraction function in extracting primary interests. DMAN-NRAN replaces the recurrent attention network with vanilla attention network to demonstrate the effectiveness of extending context for effective user modeling.

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

Experimental Settings

We obtain the behavior sequence by sorting behaviors in chronological order. Following the traditional way (Kang and McAuley 2018), we employ the last and second last interactions for testing and validation, respectively, and the remaining for training. We follow the widely-adopted way (Li et al. 2017; Lv et al. 2019) and split the ordered training sequence into L consecutive sequences. The maximum length of a sequence is T. The statistics of four datasets are listed in Table 2. We repeatedly run the model five times and report the average results.

Evaluation metrics For each user in the test set, we treat all the items that the user has not interacted with as negative items. To estimate the performance of top-K recommendations, we use Hit Rate (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K) metrics, which are widely used in the literature (He and Chua 2017).

Parameter settings For baselines, we use the source code released by the authors, and their hyper-parameters are tuned to be optimal based on the validation set. To enable a fair comparison, all methods are optimized with the number of samples equals 5 and the number of embedding dimensions D equals 128. We implement DMAN with Tensorflow and the Adam optimizer is utilized to optimize the model with learning rate equals 0.001. The batch size is set to 128 and the maximum epoch is 8. The number of memory slots m and attention layers L are searched from $\{2, 4, 6, 8, 10, 20, 30\}$ and $\{1, 2, 3, 4, 5\}$, respectively.

Comparisons with SOTA

In this section, we compare our model with different baselines. Tables 3 and 4 report the results. In general, we have three aspects of observations.

Influence of modeling long behavior sequence for traditional sequential methods From Table 3, we observe that GRU4Rec, Caser, and SASRec improve their performance when considering longer behavior sequence. Therefore, modeling longer behavior sequence has proved to be effective for user modeling. Besides, different sequential modules have varied abilities in handling long behavior sequence. Specifically, SASRec, GRU4Rec, and Caser improve 24.74%, 29.36%, and 13.42% on Taobao in terms of HR@50, while SASRec consistently performs the best. It indicates the ability of self-attention network in extracting se-

Table 3: Sequential recommendation performance over three benchmarks. $*$ indicates the model only use the latest behavior sequence for training; otherwise, the whole behavior sequence. The second best results are underlined.

Models	MovieLens			Taobao			JD.com		
	HR@10	HR@50	NDCG@100	HR@50	HR@100	NDCG@100	HR@10	HR@50	NDCG@100
GRU4Rec*	17.69	43.13	16.90	10.42	14.01	4.23	27.65	38.73	23.40
Caser*	18.98	45.64	17.62	13.71	16.51	6.89	29.27	40.16	24.25
SASRec*	21.02	47.28	19.05	16.41	22.83	9.23	33.98	44.89	27.41
GRU4Rec	19.78	47.40	18.75	13.48	16.53	5.81	35.28	47.52	27.64
Caser	20.80	48.12	19.28	15.55	17.91	7.35	36.76	49.13	28.35
SASRec	22.96	50.09	20.36	20.47	24.48	9.84	38.99	52.64	31.32
SHAN	21.34	49.52	19.55	18.87	21.94	8.73	37.72	50.55	29.80
HPMN	22.84	50.54	19.77	19.98	24.37	9.66	39.14	53.22	32.24
SDM	23.42	51.26	20.44	21.66	25.42	10.22	40.68	55.30	34.82
DMAN	**25.18**	**53.24**	**22.03**	**24.92**	**29.37**	**11.13**	**44.58**	**58.82**	**36.93**
Improv.	7.51%	3.86%	7.77%	15.05%	15.53%	8.90%	9.58%	6.36%	6.05%

Table 4: Performance on long user behavior data XLong.

Method	Recall@200	Recall@500
GRU4Rec*	0.079	0.098
Caser*	0.084	0.105
SASRec*	0.105	0.123
GRU4Rec	0.046	0.063
Caser	0.023	0.041
SASRec	0.061	0.096
SHAN	0.091	0.115
HPMN	0.118	0.136
SDM	0.107	0.129
DMAN	**0.132**	**0.163**

quential patterns, and also validates our motivation to extend self-attention network for long behavior sequence modeling.

Comparison with baselines on general datasets As shown in Table 3, our model DMAN achieves better results than baselines across three datasets. In general, long sequential models perform better than traditional sequential methods, excepting SASRec. SASRec performs better than SHAN and comparable to HPMN in most cases. This further implies the effectiveness of self-attention network in capturing long-range dependencies. The improvement of SDM over SASRec shows that explicitly extract long-term and short-term interests from long sequence is beneficial. Considering DMAN and SDM, DMAN consistently outperforms SDM over all evaluation metrics. This can be attributed to that DMAN utilizes a dynamic memory network to actively extract long-term interests into a small set of memory blocks, which is easier for the attention network to effectively attend relative information than from a long behavior sequence.

Comparison with baselines on long behavior dataset Table 4 summarizes the results of all methods on XLong, where the length of behavior sequence is larger than 1000 on average. Obviously, DMAN significantly outperforms other baselines. Compared with the findings in Table 3, one interest observation is that traditional sequential methods, i.e., GRU4Rec, Caser, and SASRec, performs poorly when directly handling long behavior sequence, and lose to long sequential models in all cases. These results demonstrate the necessity of developing new architectures tailored for long behavior sequence modeling. Another unexpected observation is that HPMN outperforms SDM on average. It further implies the ineffectiveness of attention network in attending relative messages over long sequence. By equipping attention network with the dynamic memory, our model allows us to actively update user's long-term interests in the memory and outperforms HPMN.

Table 5: Ablation study of DMAN.

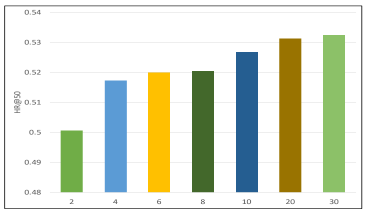
Dataset	Method	Recall@100	NDCG@100
Taobao	**DMAN-XL**	0.237	0.094
	DMAN-FIFO	0.263	0.108
	DMAN-NRNA	0.257	0.104
	DMAN	0.293	0.111
XLong	**DMAN-XL**	0.021	0.013
	DMAN-FIFO	0.036	0.017
	DMAN-NRAN	0.043	0.019
	DMAN	0.054	0.022

Ablation Study

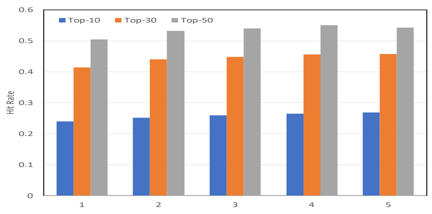
We also conduct experiments to investigate the effectiveness of several core components of the proposed DMAN. Table 5 reports the results on two representative datasets. Obviously, DMAN significantly outperforms the other three variants. The substantial difference between DMAN and DMAN-XL shows that recurrent connection is not enough to capture user's long-term interests. The improvement of DMAN over DMAN-FIFO validates that the proposed abstraction function is effective to extract user's primary long-term interests. Besides, DMAN outperforms DMAN-NRAN in general, which verifies the usefulness of extending current context with previous hidden sequence state for short-term interest extraction.

Hyper-parameter Analysis

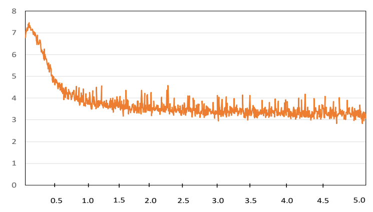
We further study the impacts of our model w.r.t. the number of memory slots m and attention layers L on Movie-



(a) Memory slots m



(b) Layer size L



(c) Learning curve

Figure 2: The proposed DMAN analysis

Lens. As we can see in Figure 2(a), DMAN achieves satisfactory results when $m = 20$ and the gain slows down with less than 2% improvement when m further increases. In experiments, we found 20 is enough for MovieLens, Taobao, JD.com and XLong. From Figure 2(b), we observe that the number of attention layers has positive impacts in our model. To trade-off between memory costs and performance, we set $L = 2$ for all datasets since it already achieves satisfactory results. Besides, we also plot the learning curve of DMAN on Taobao dataset in Figure 2(c), we can observe that DMAN converges quickly after about 2 epochs. Similar observations have been observed on other datasets. Specifically, DMAN tends to converge after 2 epochs on Taobao, JD.com and XLong datasets, while 50 epochs for MovieLens data. These results demonstrate the training efficiency of our model.

Related Work

General Recommendation

Early recommendation works largely focused on explicit feedback (Koren 2008). The recent research focus is shifting towards implicit data (Li and She 2017; Hu, Koren, and Volinsky 2008). The typical examples include collaborative filtering (Sarwar et al. 2001; Schafer et al. 2007), matrix factorization techniques (Koren, Bell, and Volinsky 2009), and factorization machines (Rendle 2010). The main challenge lies in representing users or items with latent embedding vectors to estimate their similarity. Due to their ability to learn salient representations, neural network-based models (Guo et al. 2017; Su and Khoshgoftaar 2009; Tan, Liu, and Hu 2019) are also attracted much attention recently. Some efforts adopt neural networks to extract side attributes for content-aware recommendation (Kim et al. 2016), while some aim to equip matrix factorization with non-linear interaction function (He and Chua 2017) or graph convolutional aggregation (Wang et al. 2019b; Liu et al. 2019). In general, deep learning-based methods perform better than traditional counterparts (Sedhain et al. 2015; Xue et al. 2017).

Sequential Recommendation

Sequential recommendation takes as input the chronological behavior sequence for user modeling. Typical examples belong to three categories. The first relies on temporal matrix factorization (Koren 2009) to model user's drifting preferences. The second school uses either first-order (Rendle, Freudenthaler, and Schmidt-Thieme 2010; Cheng et al. 2013) or hider-order (He and McAuley 2016; He et al. 2016; Yan et al. 2019) Markov-chains to capture the user state dynamics. The third stream applies deep neural networks to enhance the capacity of feature extraction (Yuan et al. 2019; Sun et al. 2019; Hidasi and Karatzoglou 2018). For example, Caser (Tang and Wang 2018) applies CNNs to process the item embedding sequence, while GRU4Rec (Hidasi et al. 2015) uses gated recurrent unit GRU for session-based recommendation. Moreover, SASRec (Kang and McAuley 2018) employs self-attention networks (Vaswani et al. 2017) to selectively aggregate relevant items for user modeling.

However, these methods mainly focus on making recommendations based on relatively recent behaviors. Recently, a few efforts attempt to model long behavior sequence data. For instance, SDM (Lv et al. 2019) and SHAN (Ying et al. 2018a) split the whole behavior sequence into short-term and long-term sequences and then explicitly extract long-term and short-term interest embeddings from them. But they are difficult to capture long-term interests shifting and suffer from high computation complexity. HPMN (Ren et al. 2019) uses the memory network (Graves, Wayne, and Danihelka 2014; Chen et al. 2018) to memorize important historical behaviors for next-item prediction. Nevertheless, memory network may suffer from long-term dependency forgetting dilemma, as the memory is optimized by recovering the next item. Our model focuses on combing external memory and attention networks for effective long user behavior sequence modeling, which conducts an explicit and adaptive modeling process.

Conclusions

In this paper, we propose a novel dynamic memory-based attention network DMAN for sequential recommendation with long behavior sequence. We truncate a user's overall behavior sequence into a series of sub-sequences and train our model in a dynamic manner. DMAN can explicitly extract a user's short-term and long-term interests based on the recurrent connection mechanism and a set of external memory blocks. To improve the memory fidelity, we derive a dynamic memory network to actively abstract a user's long-term interests into the memory by minimizing a local reconstruction loss. Empirical results on real-world datasets demonstrate the effectiveness of DMAN in modeling long user behavior sequences.

References

Belletti, F.; Chen, M.; and Chi, E. H. 2019. Quantifying Long Range Dependence in Language and User Behavior to improve RNNs. In *KDD*, 1317–1327.

Chen, X.; Xu, H.; Zhang, Y.; Tang, J.; Cao, Y.; Qin, Z.; and Zha, H. 2018. Sequential recommendation with user memory networks. In *WSDM*, 108–116.

Cheng, C.; Yang, H.; Lyu, M. R.; and King, I. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *RecSys*, 191–198.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* .

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* .

He, R.; Fang, C.; Wang, Z.; and McAuley, J. 2016. Vista: a visually, socially, and temporally-aware model for artistic recommendation. In *RecSys*, 309–316.

He, R.; and McAuley, J. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, 191–200. IEEE.

He, X.; and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*, 355–364.

Hidasi, B.; and Karatzoglou, A. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, 843–852.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* .

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*, 263–272. Ieee.

Jean, S.; Cho, K.; Memisevic, R.; and Bengio, Y. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* .

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *ICDM*, 197–206. IEEE.

Kim, D.; Park, C.; Oh, J.; Lee, S.; and Yu, H. 2016. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*, 233–240.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 426–434.

Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *KDD*, 447–456.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* (8): 30–37.

Li, C.; Liu, Z.; Wu, M.; Xu, Y.; Zhao, H.; Huang, P.; Kang, G.; Chen, Q.; Li, W.; and Lee, D. L. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *CIKM*, 2615–2623.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017. Neural attentive session-based recommendation. In *CIKM*, 1419–1428.

Li, J.; Wang, Y.; and McAuley, J. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*, 322–330.

Li, X.; and She, J. 2017. Collaborative variational autoencoder for recommender systems. In *GKDD*, 305–314.

Liu, N.; Tan, Q.; Li, Y.; Yang, H.; Zhou, J.; and Hu, X. 2019. Is a single vector enough? exploring node polysemy for network embedding. In *KDD*, 932–940.

Lv, F.; Jin, T.; Yu, C.; Sun, F.; Lin, Q.; Yang, K.; and Ng, W. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *CIKM*, 2635–2643.

Ma, C.; Ma, L.; Zhang, Y.; Sun, J.; Liu, X.; and Coates, M. 2019. Memory Augmented Graph Neural Networks for Sequential Recommendation. *arXiv preprint arXiv:1912.11730* .

Rae, J. W.; Potapenko, A.; Jayakumar, S. M.; and Lillicrap, T. P. 2019. Compressive Transformers for Long-Range Sequence Modelling. *arXiv preprint arXiv:1911.05507* .

Ren, K.; Qin, J.; Fang, Y.; Zhang, W.; Zheng, L.; Bian, W.; Zhou, G.; Xu, J.; Yu, Y.; Zhu, X.; et al. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*, 565–574.

Rendle, S. 2010. Factorization machines. In *ICDM*, 995–1000. IEEE.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 811–820.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *NIPS*, 3856–3866.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*, 285–295. ACM.

Schafer, J. B.; Frankowski, D.; Herlocker, J.; and Sen, S. 2007. Collaborative filtering recommender systems. In *The adaptive web*, 291–324. Springer.

Sedhain, S.; Menon, A. K.; Sanner, S.; and Xie, L. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*, 111–112.

Sodhani, S.; Chandar, S.; and Bengio, Y. 2018. On training recurrent neural networks for lifelong learning. *CoRR, abs/1811.07017* .

Su, X.; and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. In *NeurIPS*, 2440–2448.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 1441–1450.

Tan, Q.; Liu, N.; and Hu, X. 2019. Deep Representation Learning for Social Network Analysis. *Frontiers in Big Data* 2: 2.

Tan, Q.; Liu, N.; Zhao, X.; Yang, H.; Zhou, J.; and Hu, X. 2020. Learning to Hash with Graph Neural Networks for Recommender Systems. In *WWW*, 1988–1998.

Tan, Q.; Zhang, J.; Yao, J.; Liu, N.; Zhou, J.; Yang, H.; and Hu, X. 2021. Sparse-interest network for sequential recommendation. In *WSDM*.

Tang, J.; and Wang, K. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 565–573.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Wang, B.; Zhao, D.; Lioma, C.; Li, Q.; Zhang, P.; and Simonsen, J. G. 2019a. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333* .

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019b. Neural graph collaborative filtering. In *SIGIR*, 165–174.

Xue, H.-J.; Dai, X.; Zhang, J.; Huang, S.; and Chen, J. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*, volume 17, 3203–3209. Melbourne, Australia.

Yan, A.; Cheng, S.; Kang, W.-C.; Wan, M.; and McAuley, J. 2019. CosRec: 2D Convolutional Neural Networks for Sequential Recommendation. In *CIKM*, 2173–2176.

Ying, H.; Zhuang, F.; Zhang, F.; Liu, Y.; Xu, G.; Xie, X.; Xiong, H.; and Wu, J. 2018a. Sequential recommender system based on hierarchical attention network. In *IJCAI*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018b. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*, 974–983. ACM.

Yu, F.; and Koltun, V. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* .

Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J. M.; and He, X. 2019. A simple convolutional generative network for next item recommendation. In *WSDM*, 582–590.

Zhang, S.; Tay, Y.; Yao, L.; and Sun, A. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414* .

Zhu, H.; Li, X.; Zhang, P.; Li, G.; He, J.; Li, H.; and Gai, K. 2018. Learning tree-based deep model for recommender systems. In *KDD*, 1079–1088.