Trends-enhanced Attention & Memory Networks for E-commerce Recommendation

Zhi Li
li.zhi@ist.osaka-u.ac.jp
Osaka University
Japan

Daichi Amagata
amagata.daichi@ist.osaka-u.ac.jp
Osaka University
Japan

Takuya Maekawa
maekawa@ist.osaka-u.ac.jp
Osaka University
Japan

Kei Yonekawa
ke-yonekawa@kddi-research.jp
KDDI Research Inc.
Japan

Mori Kurokawa
mo-kurokawa@kddi-research.jp
KDDI Research Inc.
Japan

Takahiro Hara
hara@ist.osaka-u.ac.jp
Osaka University
Japan

ABSTRACT

How to capture both users' static and dynamic interests is a general problem for recommender systems (RSs) in e-commerce. Researchers have recently introduced neural memory recommender networks (NMRNs) to handle sparse data and capture users' dynamic interests better, which usually appear in e-commerce platforms. However, NMRNs update their memory with only the last interacted item, and this interaction-wise memory updating approach makes them too sensitive to some accidental interactions, slowing down their learning speed significantly. Motivated by this observation, we develop a market-based neural memory recommender network (MB-NMRN). We specifically propose a novel batch-wise memory updating approach that aggregates a market vector to minimize the influence of accidental interactions. In addition, we leverage a simplified pairwise hinge-loss to speed up the training of MB-NMRN. Our experimental results show that MB-NMRN achieves a remarkable improvement in recommendation performance over the current state-of-the-art methods and significantly speeds up learning compared with that of NMRN.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Recommender systems, E-commerce, Memory network, Market trends, Dynamic user preferences, Generative adversarial networks

ACM Reference Format:
Zhi Li, Daichi Amagata, Takuya Maekawa, Kei Yonekawa, Mori Kurokawa, and Takahiro Hara. 2022. Trends-enhanced Attention & Memory Networks for E-commerce Recommendation. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'22).* ACM, New York, NY, USA, 9 pages.

SIGIR eCom'22, July 15, 2022, Madrid, Spain

1 INTRODUCTION

With the explosive growth of available online information, users spend a long time to select their suitable items from many products, movies, and restaurants. Recommender systems (RSs) are an intuitive and effective choice to defend against this consumer overchoice problem [25, 26, 29, 30, 37]. Moreover, utilizing RSs becomes a general way to improve user experiences and supplier profits [1].

RSs can be categorized into collaborative filtering RSs [16] and content-based RSs [21]. Collaborative filtering RSs generate recommendation lists based on user-item interaction records. In contrast, content-based RSs are trained on the user and item side information, e.g., descriptions of items, including text, images, and videos. As content-based RSs rely on domain-specific side information, it is hard to design robust RSs. Therefore, to build a robust RS that does not rely on side information, many studies focus on collaborative filtering RSs. Traditional collaborative filtering RSs [16] first encode users and items into a high-dimensional embedding space and calculate the similarities between users and items with the user and item vectors in the embedding space. Then, these systems recommend the most similar items to users. Collaborative filtering RSs usually assume that user preferences and item attributes are static. However, information in the real world is always dynamic. For example, [34] has introduced three temporal and dynamic factors in a movie recommendation field: changes in movie perceptions, seasonal changes, and user interests. Static preference-based RSs cannot correctly process these dynamic factors, resulting in poor recommendation performance. As in the case of the movie domain, static RSs also perform poorly in e-commerce recommendations, particularly in the case of flash-sales [9]. In an e-commerce scenario, we have two observations: (i) available items and discount items frequently change and (ii) users are attracted to recent market trends, i.e., they show dynamic interest. As a result, both the dynamic and the static parts of user interests need to be considered when developing RSs for e-commerce.

Many studies introduce recurrent neural networks (RNNs) and long short-term memory (LSTM) models to capture temporal information from users' past interactions [9, 19, 34]. However, most RNN- and LSTM-based models can serve only for session-based RSs [31], which can process only time-series data. These models have limitations in capturing the stable part of users' interests and the inherent part of item attributes, because RNN and LSTM update all the memory cells at a time. In addition, session-based RSs usually

ignore users with few interactions to improve recommendation performance, because they have poor performance on such users. These RSs consequently lose their superiority over the highly sparse data, which is prevalent in the real world. Wang et al. hence developed a novel key-value memory network (KV-MemNN) [24] based RS, namely the neural memory recommender network (NMRN), to relieve the above restrictions. Different from RNNs and LSTM, KV-MemNNs can read and write part of their memories to capture both the dynamic and static parts of user interests. However, NMRNs update their memories at an interaction-wise frequency. This update strategy raises two problems: (i) The learning speed of NMRN is slow because the computation of the next interaction must wait until the memory update of the previous interaction is finished, and (ii) NMRNs update their memory with only the last interacted item, which makes NMRNs too sensitive to some accidental interactions and leads to a decrease in recommendation accuracy.

To address these problems, we develop market-based neural memory recommender networks, MB-NMRNs. Specifically, we propose a batch-wise memory updating approach and simplify the loss function of NMRNs. Users' dynamic interests are usually caused by market trends changes. Therefore, instead of applying the last interacted item to update the memory, we aggregate a market vector with some recent interacted items to represent the change of the market trends and update the memories of MB-NMRNs with this market vector. Thus, MB-NMRNs can minimize the influence of accidental interactions and accurately track the market trends to facilitate the learning of users' dynamic interests. Furthermore, updating the memory of an MB-NMRN per mini-batch enables training of the NMRN in parallel with the mini-batch trick [23], i.e., its training speed can be accelerated by matrix computation.

To summarize, this work makes the following contributions:

- We develop a market-based neural memory recommender network and propose a batch-wise memory updating approach. Also, we introduce a market vector to track the change in market trends. This improvement enables MB-NMRNs to better learn users' dynamic interests.
- We remove the penalties of positive items from the loss function of NMRNs. By combining our batch-wise memory updating approach and this simplified loss function, the training of MB-NMRN can be accelerated with a mini-batch trick.
- We conduct experiments on a real-world e-commerce dataset. The experimental results demonstrate that our model significantly outperforms NMRN and the other competitors in recommendation accuracy and computational efficiency.

Roadmap. We review some related works in Section 2, and then we introduce NMRN in Section 3. Our proposed framework is presented in Section 4.1. Section 5 reports our experimental results. Finally, Section 6 concludes this paper.

2 RELATED WORK

2.1 Collaborative Filtering RSs

Collaborative filtering RSs generate recommendations based on historical user-item interactions, including explicit (e.g., previous ratings) and implicit feedback (e.g., past purchases). One of the most famous collaborative filtering RSs is matrix factorization (MF) [16]. MF learns static latent vectors that represent users and items by optimizing the prediction and the ground truth value in the user-item interaction matrix. In recent years, inspired by the great success of deep learning and graph neural networks in many fields, developing RSs based on deep learning and graph neural networks has become a promising research direction. For example, NeuCF combines an MF and a Multi-Layer Perceptron (MLP) to improve recommendation accuracy by learning user and item vectors containing deep and shallow features. He et al. [6] proposed LightGCN to enhance user and item embeddings by extracting the high-order structure information from a user-item interaction graph with graph convolutional networks.

The above RSs ignore the temporal and dynamic information in users' interactions, and it is difficult for these systems to provide accurate recommendations. Therefore, [12, 14, 20] have proposed session-based models. They can capture sequential information from users' historical interactions by utilizing sequential models, such as RNNs and LSTM. However, they can serve only time-series data and produce degraded recommendation performances on users with few interactions.

2.2 Memory Networks

Memory-based neural networks (MemNNs) are novel learning models inspired by the advances of modern computer architecture [5]. MemNNs can flexibly manage their memories by reading and writing parts of memory components [17]. In [33], Weston et al. firstly proposed the concept of MemNNs. MemNNs share their memories among all data and can capture the sequential change of data at a distribution level. They have demonstrated that MemNNs outperform RNNs because the representation ability of MemNNs is higher than that of RNNs. MemNNs have been leveraged in many research fields, such as question answering [24, 27] and knowledge tracking [8].

In [31], Wang et al. proposed a novel neural memory recommender network (NMRN) to develop MemNN-based RSs. NMRNs can capture both users' stable and dynamic interaction patterns, and the dynamic part is learned by updating the memories with the last item information. However, NMRNs update their memories per interaction, and thus the computation of a new interaction must wait until the memory update of the previous interaction is finished. As a result, NMRNs cannot serve real-world recommendations, which require RSs with a fast computational speed to improve user experiences. In addition, updating memories with only the last interacted item makes NMRNs too sensitive to some accidental interactions and leads to a decrease in recommendation accuracy.

3 NEURAL MEMORY RECOMMENDER NETWORK

This section describe the architecture of the original NMRN, which is shown in Figure 1. NMRN is a recommender system developed based on KV-MemNN and generative adversarial networks [4], which are composed of a discriminator (D) and a generator (G). The generator of NMRN samples informative negative items to better train the discriminator. The discriminator of NMRN is a KV-MemNN that captures users' static and dynamic interaction patterns to better measure the similarities between users and items. After

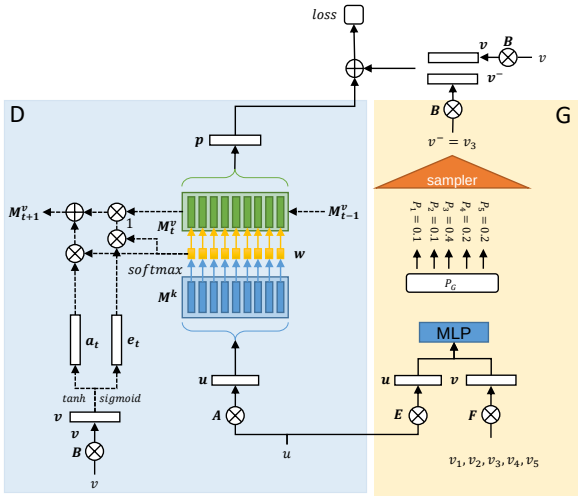


Figure 1: The architecture of NMRN. NMRN contains a discriminator, which measures similarities between users and items, and a generator, which samples informative negative items to help the learning of discriminator.

the discriminator calculates the similarities between a user and items, NMRN recommends the items with the highest similarities to the user. In this work, we denote matrices and vectors with bold capital letters and bold small letters, respectively.

3.1 Discriminator

Given a user u, an item v, and their interaction (u, v), the discriminator measures the similarity between u and v. It first calculates the latent user embedding $\mathbf{u}$ through multiplying u by a user embedding matrix $\mathbf{A}$. The latent item embedding $\mathbf{v}$ is also calculated through multiplying v by an item embedding matrix $\mathbf{B}$. The shapes of $\mathbf{A}$ and $\mathbf{B}$ are $\mathbb{R}^{N\times r}$ and $\mathbb{R}^{M\times r}$, respectively. N, M, and r are the number of users, items, and embedding dimensions, respectively. Similar to KV-MemNNs in the QA systems [24], this discriminator leverages a key memory matrix $\mathbf{M}^k$ and a value memory matrix $\mathbf{M}_t^v$ to capture both users' static and dynamic interests. $\mathbf{M}^k$ and $\mathbf{M}_t^v$ have the same shape $\mathbb{R}^{L\times r}$, where L is the number of memory slots. Each slot in $\mathbf{M}_t^v$ represents a type of latent user interaction pattern at time t, while each slot in $\mathbf{M}_k$ serves as a key to help users retrieval latent interest patterns in $\mathbf{M}_t^v$. To capture both users' static and dynamic interests, the user embedding $\mathbf{u}$ is enhanced with the latest user interaction patterns in $\mathbf{M}_t^v$. First, the similarity between user u and each slot in $\mathbf{M}_k(i)$ is measured by the inverse Euclidean distance:

$$sim_i = -\left\|\mathbf{u} - \mathbf{M}^k(i)\right\|. \tag{1}$$

Then, the softmax normalized similarities serves as the attention scores to help retrieve information from $\mathbf{M}_t^v$, where the attention score of slot i is calculated as:

$$w_i = \frac{e^{\text{sim}_i}}{\sum_{j=1}^{L} e^{\text{sim}_j}}. \tag{2}$$

After that, the memory-enhanced user vector $\mathbf{p}$ is calculated by aggregating the different latent user interaction patterns in $\mathbf{M}_t^v$ under the guidance of the above attention scores, where $\mathbf{p}$ is formulated as:

$$\mathbf{p} = \sum_{i=1}^{L} w(i)\mathbf{M}_t^v(i). \tag{3}$$

Finally, the similarity between user u and item v is defined as the inverse Euclidean distance between $\mathbf{p}$ and $\mathbf{v}$, where the Euclidean distance between $\mathbf{p}$ and $\mathbf{v}$ is calculated by:

$$d(u, v) = ||\mathbf{p} - \mathbf{v}|| = \sqrt{\sum_{i=1}^{r}(p_i - v_i)^2}. \tag{4}$$

The discriminator is optimized by minimizing a novel weighted approximate-rank pairwise (WARP) loss [10]. The WARP loss is defined as:

$$\mathcal{L} = \sum_{(u,v)\in S} \sum_{v^-\sim\mathcal{V}_u^-} \lambda_{u,v} * |q + d(u, v) - d(u, v^-)|_+, \tag{5}$$

where $|z|_+ = max(z, 0)$ denotes the standard hinge-loss, q is the safety margin size, which should be larger than zero, S is the training user-item interaction dataset, v^- is a negative item sampled by the generator, and $\mathcal{V}_u^-$ is a randomly sampled item subset from items with which u has never interacted. Besides, $\lambda_{u,v}$ denotes the penalty of a positive item v:

$$\lambda_{u,v} = \log\left(rank_{u,v} + 1\right), \tag{6}$$

where $rank_{u,v}$ is the rank of item v in u's recommendation list. Because ranking all items results in a high computational cost, NMRN ranks sampled negative items. The approximate rank is calculated as:

$$rank_{u,v} \approx \left\lfloor \frac{N-1}{N_{u,v}} \right\rfloor. \tag{7}$$

Note that N is the total number of items and $N_{u,v}$ is the number of negative items that need to be drawn until v^- satisfying $d(u, v) - d(u, v^-) + q > 0$.

3.2 Interaction-wise Memory Update

To capture both user's static and dynamic interests, the discriminator updates its value memory after the similarity calculation of each user-item interaction $(u, v) \in S$. For each $(u, v) \in S$, after $d(u, v)$ is calculated, the value memory $\mathbf{M}_t^v$ is updated by the latent embedding of item v, i.e., $\mathbf{v}$. The discriminator first erases a part of the memories and stores the user's static interests with the remaining part of memories. Then, it adds the most recent interaction information into its memories with the last interacted item. In the erasing step, the value memory matrix $\mathbf{M}_t^v$ is partially erased and modified by an *erased vector* $\mathbf{e}_t$:

$$\tilde{\mathbf{M}}_{t+1}^v(i) = \mathbf{M}_t^v(i) \circ [\mathbf{I} - w(i)\mathbf{e_t}] \tag{8}$$

$$\mathbf{e}_t = \text{Sigmoid}\left(\mathbf{W}_e\mathbf{v} + \mathbf{b}_e\right), \tag{9}$$

where $\mathbf{I}$ is a vector with all elements being 1, $\circ$ is element-wise multiplication, $\mathbf{W}_e$ is a linear transformation matrix, and $\mathbf{b}_e$ is a bias vector. In the adding step, an *add vector* $\mathbf{a}_t$ is calculated in a similar way:

$$\mathbf{a}_t = \text{Tanh}\left(\mathbf{W_a}\mathbf{v} + \mathbf{b_a}\right), \tag{10}$$

where $\mathbf{W}_a$ and $\mathbf{b}_a$ are a linear transformation matrix and bias vector, respectively. Finally, the updated value memory matrix is calculated as follows:

$$\mathbf{M}^v_{\mathbf{t+1}}(i) = \tilde{\mathbf{M}}^v_{\mathbf{t+1}}(i) + w(i)\mathbf{a_t}. \tag{11}$$

Due to this interaction-wise memory-updating operation, the similarity inference of each interaction has to wait until the previous memory updating is finished. This significantly limits the training speed of the discriminator.

3.3 Generator

The goal of the generator is to generate plausible negative items so that the discriminator faces difficulty in distinguishing interacted items and negative items. Therefore, the objective function is to maximize the expectation of the similarity between users and the generated negative items. The loss function of the generator $\mathcal{L}_G$ is described as follows:

$$\mathcal{L}_G = \sum_{\substack{(u,v)\in\mathcal{S} \\ v^-\sim P_G(v^-|u,v)}} \mathbb{E}\left[-d_D\left(u,v^-\right)\right], \tag{12}$$

where $-d_D(u,v)$ denotes the similarity between a user u and an item v measured by the discriminator. $P_G\left(v^-|u,v\right)$ is the probability of sampling a negative item v^- for a given interaction (u,v), and it is calculated as:

$$P_G\left(v^-|u,v\right) = \frac{\exp\left(-d_G\left(u,v^-\right)\right)}{\sum_{\bar{v}\in\mathcal{V}_u^-}\exp\left(-d_G(u,\bar{v})\right)}, \tag{13}$$

where $d_G(u,v)$ is the Euclidean distance between user u and item v. Note that $\mathcal{V}_u^-$ is a set of negative items $\bar{v}$ randomly sampled from items never interacted with user u. To calculate $d_G(u,v)$, the generator encodes users and items with the transformation matrices $\mathbf{E}$ and $\mathbf{F}$ and utilizes two MLPs to separately extract deep embeddings of users and items. Then, the generator measures the Euclidean distance in the above deep embedding space. The generator applies a policy gradient-based reinforcement learning algorithm to optimize its parameters based on the loss $\mathcal{L}_G$. The gradient of $\mathcal{L}_G$ is calculated as:

$$\begin{aligned}\nabla_{\theta_G}\mathcal{L}_G &= \sum_{(u,v)\in\mathcal{S}} \mathbb{E}_{v^-\sim P_G}\left[-d_D\left(u,v^-\right)\nabla_{\theta_G}\log P_G\left(v^-|u,v\right)\right] \\ &\simeq \sum_{(u,v)\in\mathcal{S}}\sum_{v_i^-\sim P_G}\left[-d_D\left(u,v_i^-\right)\nabla_{\theta_G}\log P_G\left(v_i^-|u,v\right)\right].\end{aligned} \tag{14}$$

3.4 Top-K recommendations

For each test user, the similarities between the user and negative items, which are not interacted with this user, are measured by the discriminator. Then, the K most similar items are recommended to this user.

4 MB-NMRN: MARKET-BASED NEURAL MEMORY RECOMMENDER NETWORK

This section presents the details of our proposed batch-wise memory updating approach, which enables NMRN to be more flexible to adjust its memory updating frequency. By utilizing the batch-wise memory updating approach and a simplified hinge-loss, our MB-NMRN can accelerate its learning with the mini-batch trick [23]. Table 1 summarizes the notations used in this section.

Table 1: Summary of notations

Notation	Description
$\mathbf{u}_{bh}$	Mini-batch of users
$\mathbf{v}_{bh}$	Mini-batch of items
S_{bh}	Mini-batch of interactions
H	Mini-batch size
$\mathbf{U}_{bh}$	Embedding matrix for $\mathbf{u}_{bh}$
$\mathbf{V}_{bh}$	Embedding matrix for $\mathbf{v}_{bh}$
$\mathbf{W}_{bh}$	Mini-batch attention
$\mathbf{P}_{bh}$	Deep embedding matrix for $\mathbf{u}_{bh}$
$\mathbf{C}^-_{bh}$	Negative item matrix
J	Number of negative items per user
$\mathbf{m}$	Market vector
$\mathbf{w}^m$	Market attention

4.1 Batch-wise Memory Updating

Instead of updating the value memory $\mathbf{M}^v_t$ in an interaction-wise manner, we aggregate the items from a mini-batch of interactions to learn a *market vector*, which can better represent recent interaction patterns than a single item. Then, we update $\mathbf{M}^v_t$ with this market vector, after the similarity inference of the above mini-batch of interactions. By doing so, our MB-NMRN can minimize the influence from some accidental interactions and accurately track the market trends to facilitate the learning of users' dynamic interests. In addition, mini-batch-level memory updates enable the discriminator to infer the similarities of user-item pairs in a mini-batch in parallel.

Figure 2 gives the architecture of MB-NMRN. In this figure, $\mathbf{u}_{bh}$ and $\mathbf{v}_{bh}$ respectively denote the users and the items from a mini-batch of interactions S_{bh}. The size of S_{bh} is denoted as H. The user embedding $\mathbf{U}_{bh}$ and the item embeddings $\mathbf{V}_{bh}$ are calculated through multiplying $\mathbf{u}_{bh}$ by user embedding matrix $\mathbf{A}$ and multiplying $\mathbf{v}_{bh}$ by item embedding matrix $\mathbf{B}$, respectively. Without memory updating, all users in $\mathbf{u}_{bh}$ can be enhanced with the same $\mathbf{M}^v_t$. Therefore, the discriminator can compute the mini-batch attention $\mathbf{W}_{bh}$, the deep user embeddings $\mathbf{P}_{bh}$, and the similarities $-d_D(\mathbf{u}_{bh},\mathbf{v}_{bh})$ in parallel. $\mathbf{m}$ denotes the *market vector* and $\mathbf{C}^-_{bh} \in \mathbb{R}^{H\times J}$ denotes a negative item matrix, where each row of $\mathbf{C}^-_{bh}$ is the randomly sampled candidate negative items for a user. J denotes the number of a user's candidate negative items.

Inspired by the great successes of convolutional neural networks and attention mechanisms in information aggregation, we develop two approaches to aggregating $\mathbf{m}$: (1) a one-dimensional convolutional aggregation approach and (2) a value memory-guided multi-head attention aggregation approach. Note that applications can select one of the two approaches based on their requirements.

One-dimensional convolutional aggregation. This approach learns $\mathbf{m}$ by aggregating $\mathbf{V}_{bh}$ with a one-dimensional convolutional (1Dconv) layer:

$$\mathbf{m} = \mathbf{b_c} + \text{1Dconv}\left(\mathbf{w}_c, \mathbf{V}_{bh}\right), \tag{15}$$

where $\mathbf{b}_c$ and $\mathbf{w}_c$ are the bias and weight vectors of the convolutional layer, respectively. The attentions of different items $\mathbf{W}_{bh}$ are aggregated to form a market attention $\mathbf{w}^m$. This is used to update

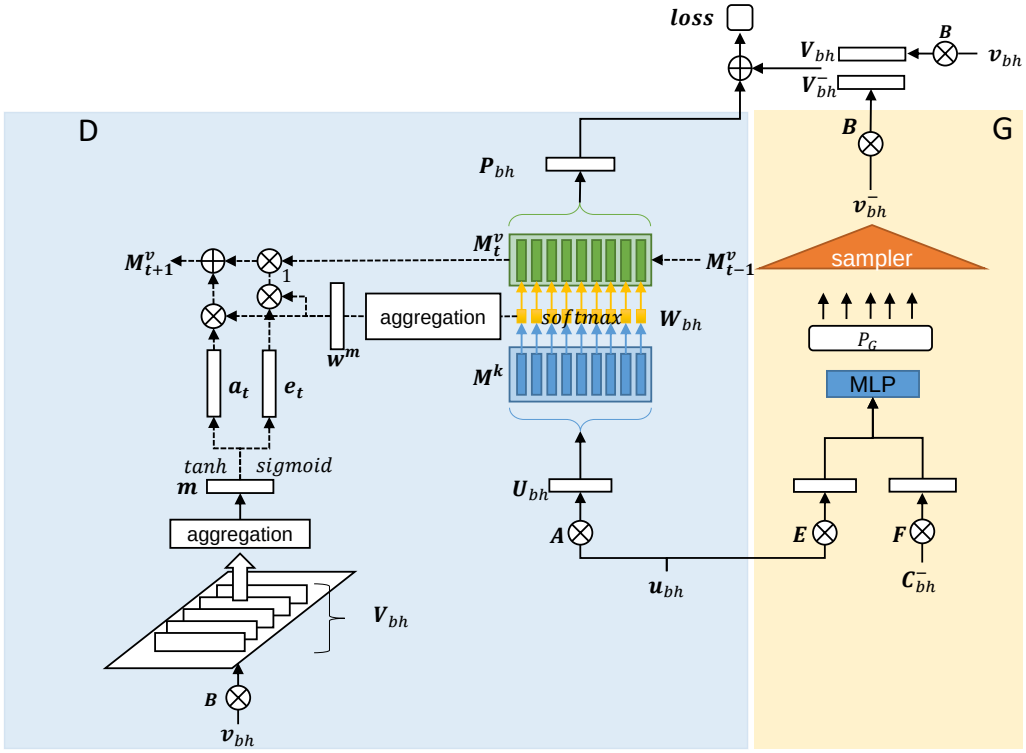


Figure 2: The architecture of MB-NMRN, which computes a mini-batch of user-item interactions in parallel

the value memory later, and $\mathbf{w}^m$ is calculated by:

$$\mathbf{w}^m = \mathbf{b}_c + \text{1Dconv}\left(\mathbf{w}_c, \mathbf{W}_{bh}\right), \tag{16}$$

where $\mathbf{m}$ and $\mathbf{w}^m$ are aggregated by the same 1Dconv layer to ensure the consistency of items and their corresponding attentions. This approach uses only a 1Dconv layer and thus can aggregate information efficiently.

Value memory-guided multi-head attention aggregation. This approach learns $\mathbf{m}$ by modeling item importance in representing recent interaction patterns. Motivated by the achievement of attention mechanism in advancing recommendations [3, 28], we introduce the attention mechanism to achieve the item importance modeling purpose. In addition, since the *market vector* is aggregated to updating the value memory, we guide the calculation of attention with the value memory and develop a novel value memory-guided multi-head attention aggregation approach. Specifically, we denote α_v^i as the attention weight of item v and the value memory slot $\mathbf{M}_t^v(i)$, where α_v^i is formulated by:

$$\begin{aligned} \alpha_v^i &= \frac{e^{\alpha_v^i}}{\sum_{v=1}^{H} e^{\alpha_v^i}}, \\ \alpha_v^i &= \mathbf{w}_1^T \,\text{ReLU}\left(\mathbf{v}_v \circ \mathbf{M}_t^v(i)\right) + b_1. \end{aligned} \tag{17}$$

$\mathbf{w}_1 \in \mathbb{R}^r$ and b_1 denote the weight and bias of a fully connected layer. $\mathbf{v}_v$ is the embedding of item v. Then, the market vector $\mathbf{m}$ can be aggregated by the mini-batch of items and their corresponding attentions:

$$\mathbf{m} = \frac{1}{L}\sum_{i=1}^{L}\sum_{v=1}^{H} \alpha_v^i \mathbf{v}_v. \tag{18}$$

The market attention $\mathbf{w}^m$ is formulated by:

$$\mathbf{w}^m = \frac{1}{L}\sum_{i=1}^{L}\sum_{v=1}^{H} \alpha_v^i \mathbf{W}_{bh}(v). \tag{19}$$

Although the attention approach takes more computation time than the one-dimensional convolutional approach, the attention approach better models the importance of different items. In Section 5, we investigate this trade-off.

After aggregating m and $\mathbf{w}^m$, the value memory matrix $\mathbf{M}_t^v$ is updated by erasing and adding the steps in Equations 8, 9, 10, and 11. The following equations show this updating process:

$$\begin{aligned} \tilde{\mathbf{M}}_{t+1}^v(i) &= \mathbf{M}_t^v(i) \circ \left[\mathbf{I} - w^m(i)\mathbf{e}_\mathbf{t}\right], \\ \mathbf{e}_t &= \text{Sigmoid}\left(\mathbf{W}_e\mathbf{m} + \mathbf{b}_e\right), \\ \mathbf{M}_{\mathbf{t+1}}^v(i) &= \tilde{\mathbf{M}}_{\mathbf{t+1}}^v(i) + w^m(i)\mathbf{a}_\mathbf{t}, \\ \mathbf{a}_t &= \text{Tanh}\left(\mathbf{W}_\mathbf{a}\mathbf{m} + \mathbf{b}_\mathbf{a}\right). \end{aligned} \tag{20}$$

4.2 Simplified Pairwise Hinge-Loss

The mini-batch trick has been introduced into deep neural networks to accelerate their training. However, existing MemNNs [24, 31, 33] cannot apply this mini-batch trick due to their *interaction-wise* memory updating operations. To address this problem, we propose

Algorithm 1: Adversarial training

Input: Training set S
Output: Parameters of discriminator θ_D, Parameters of generator θ_G

Randomly initialize parameters θ_D and θ_G
Rearrange S in chronological order
while $\mathcal{L}$ *not converged* **do**
  for *each mini-batch* $S_{bh} \in S$ **do**
    Randomly sample $\mathbf{C}^-_{bh}$
    Sample $\mathbf{v}^-_{bh}$ with $P_G(v^-|u,v)$ calculated by Eq. (13)
    Evaluate $\mathcal{L}_G$ by Eq. (12)
    Evaluate $\mathcal{L}$ by Eq. (21)
    Adversarially optimize θ_D and θ_G by $\theta_D \leftarrow \theta_D - \nabla_{\theta_D}\mathcal{L}$, $\theta_G \leftarrow \theta_G - \nabla_{\theta_G}\mathcal{L}_G$
  end
end

a batch-wise memory updating approach and a simplified pairwise hinge-loss.

Specifically, the batch-wise memory updating approach allows the discriminator of MB-NMRN to calculate the user embeddings $\mathbf{P}_{bh}$ and the similarities $-d_D(\mathbf{u}_{bh}, \mathbf{v}_{bh})$ in batch-wise manner. Note that this batch-wise manner enables parallel computation. The WARP loss requires the generator to continue sampling negative items for every interaction until $d(u,v) - d(u,v^-) + p > 0$. This requirement significantly limits the training efficiency of the discriminator. Therefore, we simplify the original WARP loss to sample only one negative item for one interaction, this makes the discriminator of MB-NMRN calculate $-d_D(\mathbf{u}_{bh}, \mathbf{v}^-_{bh})$ in batch-wise manner. The simplified pairwise hinge-loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \sum_{(\mathbf{u}_{bh}, \mathbf{v}_{bh}) \in S, \mathbf{v}^-_{bh} \sim C^-_{bh}} [q + \mathbf{d}(\mathbf{u}_{bh}, \mathbf{v}_{bh}) - \mathbf{d}\left(\mathbf{u}_{bh}, \mathbf{v}^-_{bh}\right)]_+. \quad (21)$$

As a result, MB-NMRN is more computationally efficient than the original NMRN and can better capture user's static and dynamic interests to facilitate the recommendation performance.

4.3 Training Algorithm

MB-NMRN leverages the batch-wise memory updating approach and the simplified pairwise hing-loss to optimize its discriminator parameters θ_D and generator parameters θ_G. The details of the training algorithm are summarized in Algorithm 1.

5 EXPERIMENT

The objective of our experiments is to answer the following research questions:

RQ1: How does our proposed method perform on recommendations compared with state-of-the-art methods?

RQ2: How does our proposed method perform on computational efficiency compared with NMRN?

RQ3: How do some hyper-parameters affect the recommendation accuracy of our proposed method?

Table 2: Basic information on the datasets we used

Dataset	Training	Validation	Test
#users	88,267	1,382	1,551
#items	7,121	553	499
#interactions	163,959	1,773	1,944
#avg. interaction/user	1.85	1.28	1.25

5.1 Dataset

We used a real-world e-commerce platform dataset that provides items and services (e.g., home electronics, make-up, and travel services)[1]. This dataset contains users' purchase records from *11/5/2017* to *11/6/2017*. We divided the purchase records into three parts: *11/5/2017* to *9/6/2017* as a training set, *10/6/2017* as a validation set, *11/6/2017* as a test set. For new users and items which have no interaction information in the training set, we deleted these new users and items from validation and testing sets. Some basic information on the dataset is summarized in Table 2.

5.2 Setting

5.2.1 Evaluation Criteria. For each user in the test set, we first randomly sampled 1000 items, including negative items and items that this user interacted with in the test set, where negative items indicate items that did not interact with this user in the training set.

Then, the top-k recommendations were produced by the method mentioned in Section 3.4. The result of the top-K recommendations was measured by the widely used *Hit Ratio* (HR) [2, 11, 31, 32] and *Normalized Discounted Cumulative Gain* (NDCG) [13].

Hit Ratio. Given a user-item interaction in the test set, HR@K measures whether the item is in the top-K recommendation list or not. If the target item appears in the top-K recommendation list, we obtain a *hit*. HR@K is calculated as follows:

$$HR@K = \frac{\text{Number of hits@}K}{|R|}, \quad (22)$$

where $|R|$ is the number of interactions in the test set.

Normalized Discounted Cumulative Gain. The ranking quality of the recommendation list is usually evaluated with NDCG, which accounts for the position of the *hits* by assigning higher scores to the *hits* at top ranks and downgrading the scores to *hits* at lower ranks. NDCG@K is defined as:

$$NDCG@K = \frac{DCG@K}{iDCG@K}, \qquad DCG@K = \sum_{i=1}^{K} \frac{2^{r_i} - 1}{\log_2(i+1)}, \quad (23)$$

where r_i shows the graded relevance of the target item at position i: $r_i = 1$ if the target item is ranked at the i-th position, otherwise $r_i = 0$. iDCG@K is the ideal order of DCG.

5.2.2 Evaluation methods. The experiments evaluated the following recommendation methods.

- **Top-Pop** recommends the most popular items to users.

[1] Actually, this is our private dataset.

Table 3: Hyper-parameter settings of MB-NMRN

	Hyper-parameter	Value
r	Embedding dimension	32
L	# of memory slots	64
J	# of candidate neg. items	100
q	Safety margin size	5
H	Mini-batch size	2048

- **NeuCF** [7] jointly learns a neural network and a matrix factorization model to fuse both the shallow and deep features in users' interaction patterns.
- **LightGCN** [6] develops a light graph convolutional networks to enhance the user and item embeddings with the learned structural information from the user-item interaction graph.
- **NMRN** [31] combines memory networks and generative adversarial networks to capture both users' dynamic and static interests.
- **MB-NMRN-conv**. This is our proposed method with the one-dimensional convolutional aggregation approach.
- **MB-NMRN-attn**. This is our proposed method with the value memory-guided multi-head attention aggregation approach.

5.2.3 Implementation details. The codes of NeuCF[2] and LightGCN[3] were obtained from the corresponding GitHub repositories. All evaluated methods were implemented based on PyTorch[4] framework. The optimizer was Adam [15], and the learning rate was 0.001. The other parameters were randomly initialized from a Gaussian distribution $\mathcal{N}(0, 0.01^2)$. To determine the best hyper-parameters, we tuned hyper-parameters based on the validation set. Instead of randomly sampling mini-batch user-item interactions from training set, which is applied in [31], we generated mini-batch in chronological order strictly, to mimic the interaction pattern on real-world e-commerce platforms. MB-NMRN achieves its best performance with the hyper-parameters shown in Table 3.

5.3 Comparison Results

This section reports the comparison results for recommendation performance and computational efficiency.

5.3.1 Recommendation performance. Table 4 reports the comparison results of all evaluation methods on HR@K and NDCG@K. This table shows that both MB-NMRN-conv and MB-NMRN-attn outperform all the competitors. Specifically, MB-NMRN-attn gains a 51.66% improvement on HR@1, 54.16% on HR@3, and 52.41% on HR@10 against the best competitor (LightGCN). Also, it achieves an improvement of 50.15% on NDCG@1, 49.98% on NDCG@3, and 48.9% on NDCG@10 against LightGCN. This finding empirically demonstrates that our MB-NMRN can better track the market trends to learn users' dynamic interests.

5.3.2 Computational Efficiency. To evaluate the effectiveness of our MB-NMRN on computational efficiency, we compared the run time of NMRN and MB-NMRN, where the run time indicates the time to train a model until the discriminator loss converges on the validation set. All methods run on the same Linux server with 64 Intel(R) Xeon(R) Platinum 8375C CPUs for a fair comparison. We test the mini-batch size H in $[256, 512, 1024, 2048]$. To avoid the influence of the generator, we sampled negative items uniformly at random for each interaction.

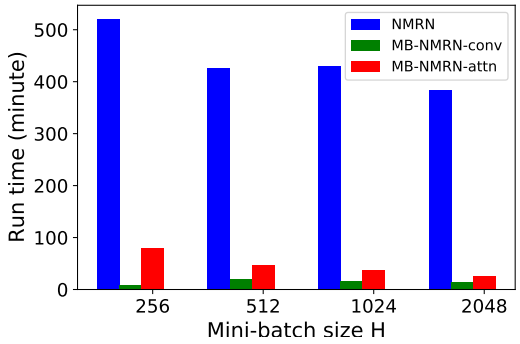
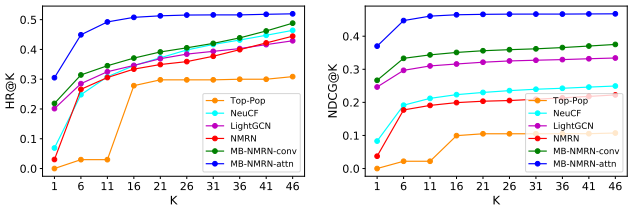


Figure 3: Run time of NMRN and MB-NMRN



(a) Performance on HR@K (b) Performance on NDCG@K

Figure 4: Impact of K

Figure 3 illustrates the run times of NMRN and MB-NMRN. We can see that both MB-NMRN-attn and MB-NMRN-conv remarkably outperform NMRN in terms of computational efficiency. This result demonstrates that our batch-wise memory updating approach and the simplified pairwise hinge-loss are effective in speeding up the training. Moreover, MB-NMRN-conv takes less run time than MB-NMRN-attn due to its light structure (i.e., a one-dimensional convolutional layer).

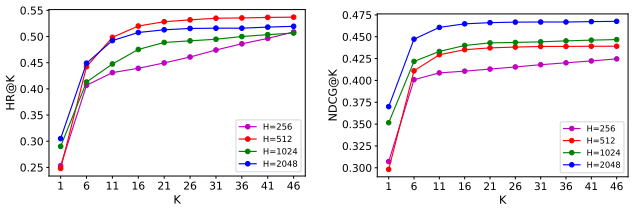
5.4 Impact of Some Hyper-Parameters

5.4.1 Recommendation List Size. To investigate the impact of the recommendation list size K, we conducted experiments by varying K. Figure 4 shows the comparison results on HR@K and NDCG@K. From this figure, we observe that MB-NMRN-attn, MB-NMRN-conv, and LightGCN achieve higher values on HR@K than the other, when K is small. This result is consistent with the result on NDCG@K, which emphasizes the importance of the head of recommendation list. This result empirically demonstrates the strength of these methods in providing accurate recommendations. We find that the HR@K of most methods achieves about 0.5 to 0.6 when K is 46. This finding indicates that ignoring market trends or confusion caused by accidental interactions can lead to mis-recommendations

[2] github.com/yihong-chen/neural-collaborative-filtering
[3] github.com/gusye1234/LightGCN-PyTorch
[4] https://pytorch.org

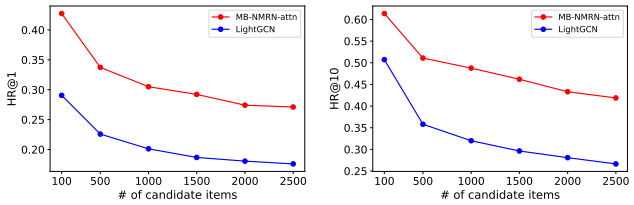
Table 4: Comparison between our proposal and the state-of-the-art by using HR@K and NDCG@K.

Method	HR@1	HR@3	HR@10	NDCG@1	NDCG@3	NDCG@10
Top-Pop	0.0000	0.0298	0.0298	0.0000	0.0220	0.0220
NeuCF	0.0689	0.1692	0.2984	0.0830	0.1536	0.2091
LightGCN	0.2011	0.2536	0.3200	0.2464	0.2826	0.3086
NMRN	0.0303	0.1615	0.2994	0.0373	0.1268	0.1891
MB-NMRN-conv	0.2186	0.3050	0.3380	0.2667	0.3287	0.3413
MB-NMRN-attn (ours)	**0.3050**	**0.3909**	**0.4877**	**0.3700**	**0.4238**	**0.4595**



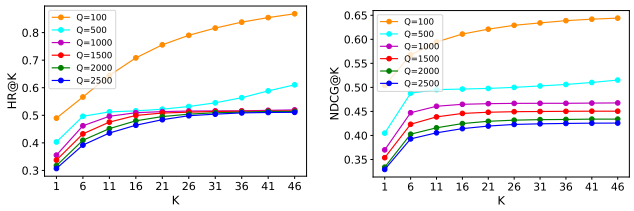
(a) Performance on HR@K (b) Performance on NDCG@K

Figure 5: Impact of H



(a) Performance on HR@1 (b) Performance on HR@10

Figure 6: Impact of candidate item number



(a) Performance on HR@K (b) Performance on NDCG@K

Figure 7: Impact of candidate item number

at the head of recommendation lists. Therefore, RSs that can accurately track market trends are required to improve recommendation performances.

5.4.2 Mini-batch Size. To study the impact of the mini-batch size H, we conducted a comparison experiment with MB-NMRN-attn trained with different H. The comparison results on HR@K and NDCG@K are illustrated in Figure 5. (We omit the result of MB-NMRN-conv, as MB-NMRN-att shows better accuracy.) In these figures, we can see that the MB-NMRN trained by a larger mini-batch performs better when K is small. This observation empirically demonstrates that fusing more recent items can more accurately capture the market trends. It is worth mentioning that NMRN equals MB-NMRN when H is set to 1, and this is the reason why NMRN performs worse than MB-NMRN.

5.4.3 Candidate Item Number. Our comparison experiment ranked 1000 candidate items for each user to produce top-K recommendations. In this part, we compared our MB-NMRN-attn (the best proposal) and LightGCN (the best baseline) by varying the number of candidate items to discuss the robustness of MB-NMRN's recommendation performance, where the number of candidate items is denoted as Q. (Since the recommendation performance of MB-NMRN-conv is similar to that of LightGCN, we omitted it.)

Figures 6 and 7 illustrate the results. From Figure 6, we observe that MB-NMRN-attn stably outperforms LightGCN on HR@1 and HR@10. Besides, we find that HR@K tends to converge when Q is larger than 1000 from Figure 7. The above results demonstrate the robustness of our MB-NMRN.

6 CONCLUSION

In this work, we developed market-based neural memory recommender networks (MB-NMRNs) for e-commerce recommendations. Specifically, we proposed a novel batch-wise memory updating approach to better capture users' dynamic interests by minimizing the influence of accidental interactions. We furthermore introduced a simplified pairwise hinge-loss to train our MB-NMRN in batch-wise parallel. The experimental results demonstrate the superiority of MB-NMRN on both recommendation performance and computational efficiency.

In e-commerce recommendations, the frequent change of available items makes the interaction information sparser, which significantly impairs recommendation performance. Therefore, we will focus on addressing the sparse data problem by developing cross-domain recommender systems that transfer knowledge from auxiliary domains to enrich the sparse interactions in the target domain [18, 22, 35, 36].

ACKNOWLEDGMENTS

This research is partially supported by JST CREST Grant Number JPMJCR21F2.

REFERENCES

[1] Daichi Amagata and Takahiro Hara. 2021. Reverse Maximum Inner Product Search: How to efficiently find users who would like to buy my item?. In *RecSys*. 273–281.

[2] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*. 39–46.

[3] Chen Gao, Xiangning Chen, Fuli Feng, Kai Zhao, Xiangnan He, Yong Li, and Depeng Jin. 2019. Cross-domain Recommendation Without Sharing User-relevant Data. In *WWW*. 491–502.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.

[5] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).

[6] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[8] Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969* (2016).

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

[10] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *WWW*. 193–201.

[11] Bo Hu and Martin Ester. 2013. Spatial topic modeling in online social media for location recommendation. In *RecSys*. 25–32.

[12] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys*. 306–310.

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[14] How Jing and Alexander J Smola. 2017. Neural survival recommender. In *WSDM*. 515–524.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[17] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*. 1378–1387.

[18] Mori Kurokawa, Hao Niu, Kei Yonekawa, Arei Kobayashi, Daichi Amagata, Takuya Maekawa, and Takahiro Hara. 2018. Virtual touch-point: trans-domain behavioral targeting via transfer learning. In *IEEE Big Data*. 4762–4767.

[19] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *KDD*. 1734–1743.

[20] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*. 1831–1839.

[21] Zheng Liu, Xing Xie, and Lei Chen. 2018. Context-aware academic collaborator recommendation. In *KDD*. 1870–1879.

[22] Yuan Lyu, Daichi Amagata, Takuya Maekawa, Takahiro Hara, Hao Niu, Kei Yonekawa, and Mori Kurokawa. 2019. Behavior Matching between Different Domains based on Canonical Correlation Analysis. In *ECNLP*. 361–366.

[23] Dominic Masters and Carlo Luschi. 2018. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612* (2018).

[24] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* (2016).

[25] Jun Murao, Kei Yonekawa, Mori Kurokawa, Daichi Amagata, Takuya Maekawa, and Takahiro Hara. 2021. Concept Drift Detection with Denoising Autoencoder in Incomplete Data. In *MobiQuitous*. 541–552.

[26] Duc Nguyen, Hao Niu, Kei Yonekawa, Mori Kurokawa, Chihiro Ono, Daichi Amagata, Takuya Maekawa, and Takahiro Hara. 2020. On the Transferability of Deep Neural Networks for Recommender System. In *IAL*. 22–37.

[27] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*. 2440–2448.

[28] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[29] Hanxin Wang, Daichi Amagata, Takuya Maekawa, Takahiro Hara, Hao Niu, Kei Yonekawa, and Mori Kurokawa. 2019. Preliminary investigation of alleviating user cold-start problem in e-commerce with deep cross-domain recommender system. In *ECNLP*. 398–403.

[30] Hanxin Wang, Daichi Amagata, Takuya Makeawa, Takahiro Hara, Niu Hao, Kei Yonekawa, and Mori Kurokawa. 2020. A DNN-Based Cross-Domain Recommender System for Alleviating Cold-Start Problem in E-Commerce. *IEEE Open Journal of the Industrial Electronics Society* 1 (2020), 194–206.

[31] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural memory streaming recommender networks with adversarial training. In *KDD*. 2467–2475.

[32] Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. 2015. Geo-SAGE: A geographical sparse additive generative model for spatial item recommendation. In *KDD*. 1255–1264.

[33] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* (2014).

[34] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.

[35] Kei Yonekawa, Hao Niu, Mori Kurokawa, Arei Kobayashi, Daichi Amagata, Takuya Maekawa, and Takahiro Hara. 2019. Advertiser-Assisted Behavioral Ad-Targeting via Denoised Distribution Induction. In *IEEE Big Data*. 5611–5619.

[36] Kei Yonekawa, Hao Niu, Mori Kurokawa, Arei Kobayashi, Daichi Amagata, Takuya Maekawa, and Takahiro Hara. 2019. A heterogeneous domain adversarial neural network for trans-domain behavioral targeting. In *PAKDD*. 274–285.

[37] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM CSUR* 52, 1 (2019), 5.