Hierarchical Gating Networks for Sequential Recommendation

Chen Ma
School of Computer Science
McGill University
chen.ma2@mail.mcgill.ca

Peng Kang
Department of Computer Science
Northwestern University
pengkang2022@u.northwestern.edu

Xue Liu
School of Computer Science
McGill University
xueliu@cs.mcgill.ca

ABSTRACT

The chronological order of user-item interactions is a key feature in many recommender systems, where the items that users will interact may largely depend on those items that users just accessed recently. However, with the tremendous increase of users and items, sequential recommender systems still face several challenging problems: (1) the hardness of modeling the long-term user interests from sparse implicit feedback; (2) the difficulty of capturing the short-term user interests given several items the user just accessed. To cope with these challenges, we propose a hierarchical gating network (HGN), integrated with the Bayesian Personalized Ranking (BPR) to capture both the long-term and short-term user interests. Our HGN consists of a feature gating module, an instance gating module, and an item-item product module. In particular, our feature gating and instance gating modules select what item features can be passed to the downstream layers from the feature and instance levels, respectively. Our item-item product module explicitly captures the item relations between the items that users accessed in the past and those items users will access in the future. We extensively evaluate our model with several state-of-the-art methods and different validation metrics on five real-world datasets. The experimental results demonstrate the effectiveness of our model on Top-N sequential recommendation.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Sequential Recommendation; Feature Gating; Instance Gating; Item-item Product

ACM Reference Format:
Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical Gating Networks for Sequential Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330984

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330984

1 INTRODUCTION

As the Internet service and mobile device usages keep growing, Internet users can easily access a large number of online products and services. Although this growth provides users with more available choices, it is also difficult for users to pick up one of the most favorite items out of plenty of candidates. To reduce information overload and satisfy the diverse needs of users, personalized recommender systems come into being and play more and more important roles in modern society. These systems can provide personalized experiences, serve huge service demands, and bring significant benefits to at least two parties: (1) help users easily discover products that they are interested in; (2) create opportunities for product providers to increase the revenue.

In all kinds of Internet services, users access the products or items in a chronological order, where the items a user will interact may be closely relevant to those items she just accessed. This property facilitates a non-trivial recommendation task—sequential recommendation, which treats the user behavior history as an action sequence ordered by the operating timestamp. This task is challenging to address due to one major reason: the difficulty of inferring users' short-term interests and intentions. Indeed, both the long-term and short-term interests of users together determine the users' actions on items. With the large accumulated data, the long-term user interests can be effectively modeled. However, within a short-term context, how to take advantage of the sequential dynamics for predicting user actions in the near future is non-trivial.

To capture the sequential dynamics in the user action history, effective models are proposed to learn the short-term user preference in the sequential user interactions, such as Markov Chains (MCs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). MC-based methods [8, 30] apply a K-order Markov chain to make recommendations based on the K previous actions. CNN-based methods [34] utilize convolutional filters and sliding window strategies to capture the short-term contexts for future prediction. RNN-based methods [12, 13, 28] adopt gated recurrent (GRU) or long short-term memory (LSTM) units to learn the user-item sequence, where the short-term user interests are captured by the hidden states of RNNs.

Although existing methods have proposed effective models and achieved satisfactory results, we argue that there are still several factors to be considered for enhancing the performance. First, previous studies [12, 13, 28, 34] learn the user action sequence by CNN or RNN structures, which does not consider the specific parts of features of different items. Neglecting the representative features may fail to capture the true user interests in a short context. Second, these CNN or RNN based methods also do not discriminate the item importance based on users' preferences. Equally treating those informative items along with other items may lead to the incomplete understanding of user intentions. Third, it is also important to note

that the relations between items are neglected in previous works [12, 18, 34]. It is very likely that closely related items may be interacted by users one after the other. As such, explicitly capturing the item-item relations will largely benefit predicting subsequent items users will interact.

To address the problems mentioned above, we propose a novel recommendation model, hierarchical gating network (HGN), for the sequential recommendation task without using complex recurrent or convolutional neural networks. HGN consists of a feature gating module, an instance gating module, and an item-item product module, integrated with the matrix factorization model and optimized by the Bayesian Personalized Ranking (BPR) objective. In particular, the feature gating module allows the adaptive selections of attractive latent features of items based on the user preference, where the selected user-specific features will be passed to the instance gating module. At the instance gating module, important items that reflect the short-term user interests will be distinguished and selected for future item prediction. Thus, the feature gating and instance gating modules form a hierarchical gating network to control what features or items can be passed to the downstream layers. On the other hand, item-item relations provide important auxiliary information to predict users' sequential behaviors, since closely related items may be interacted by users one after the other. Thus, we apply an item-item product module to explicitly capture the relations between the items users have interacted and those items user will interact in the future. We extensively evaluate our model with many state-of-the-art methods and different validation metrics on five real-world datasets. The experimental results not only demonstrate the improvements of our model over other baselines but also show the effectiveness of the gating and item-item product modules.

To summarize, the major contributions of this paper are listed as follows:

- To infer the user interests in a short-term context, we propose a hierarchical gating network to control what item latent features and which relevant item can be passed to the downstream layers. Our hierarchical gating network achieves better performance compared with complex recurrent or convolutional neural networks yet with fewer parameters and faster training speed.
- To explicitly capture the item-item relations, we utilize an item-item product module to learn the relationships between the items users have interacted and those items user will interact in the near future.
- Experiments on five real-world datasets show that the proposed HGN model significantly outperforms the state-of-the-art methods for the sequential recommendation task.

2 RELATED WORK

In this section, we illustrate related work about the proposed model: personalized recommendation with user implicit feedback and the sequential recommendation.

2.1 Recommendation with Implicit Feedback

In many real-world recommendation scenarios, user implicit data [35, 39], e.g., browsing or clicking history, is more ubiquitous and common than the explicit feedback [31, 32], such as user ratings. The implicit feedback only provides positive samples, which is also called one-class collaborative filtering (OCCF) [26]. Effective methods are proposed to tackle the OCCF problem. Early works either apply a uniform weighting scheme to treat all missing data as negative samples [15], or sample negative instances from missing data to learn the pair-wise user preference between positive and negative samples [29]. Recently, several works [11, 22] are proposed to weigh the missing data. In [14, 36], metric learning is applied to compute the distance between users and items. With the ability to represent non-linear and complex data, (deep) neural networks have been utilized in the domain of recommendation and bring more opportunities to reshape the conventional recommendation architectures. In [23, 24, 40], (denoising) autoencoders are proposed capture the user-item interaction from user implicit feedback. In [10], He et al. propose a neural network-based collaborative filtering model, where a multi-layer perceptron is utilized to learn the non-linear user-item interactions. In [5, 21, 41], conventional matrix factorization and factorization machine methods are also benefited by the representation ability of deep neural networks.

2.2 Sequential Recommendation

Some early sequential recommendation methods rely on item-item transition matrices to capture the sequential patterns in the user interaction sequence. The Markov chain [2] is a classical option to solve this problem. For example, Rendle et al. [30] propose to factorize personalized Markov chains for capturing long-term preferences and short-term transitions. He et al. [8] combines similarity-based models with high-order Markov chains to make personalized sequential recommendations. In [7], the translation-based method is proposed for sequential recommendation. Recently, benefited by the advantages of sequence learning in natural language processing, (deep) neural network based methods are proposed to learn the sequential dynamics. For instance, Tang et al. [34] propose to apply the convolutional neural network (CNN) on item embedding sequence, where the short-term contexts can be captured by the convolutional operations. In [12, 13, 20, 28], recurrent neural network (RNN), especially gated recurrent unit (GRU), based methods are utilized to model the sequential patterns for the session-based recommendation [13], where the hidden states of RNNs reflect the summary of the (sub)sequence. On the other hand, self-attention [37] exhibits promising performance in sequence learning and is utilized in sequential recommendation. In [18], Kang et al. propose to leverage self-attention for adaptively considering interacted items. In [1, 16], memory networks [33] are adopted to memorize the important items that will play a role in predicting future user actions.

However, our hierarchical gating network is different from the above studies. We apply feature-level and instance-level gating modules to adaptively control what item latent features and which relevant item can be passed to the downstream layers. While previous works either do not consider the representative items, or only consider the instance-level importance by the attention model but neglecting the feature-level ones. On the other hand, we adopt an item-item product to explicitly capture the relations between the items users have interacted and those items users will access in the near future, where the explicit modeling of item relations is rarely considered in previous works.

3 PROBLEM FORMULATION

The recommendation task considered in this paper takes sequential implicit feedback as training data. The user preference is presented by a user-item sequence in the chronological order $\mathcal{S}^i = (\mathcal{S}^i_1, \mathcal{S}^i_2, ..., \mathcal{S}^i_{|\mathcal{S}^i|})$, where $\mathcal{S}^i_j$ is an item index that user i has interacted with. Given the earlier subsequence $\mathcal{S}^i_{1:t} (t < |\mathcal{S}^i|)$ of M users, the problem is to recommend a list of items from N items to each user and evaluate whether the items in $\mathcal{S}^i_{t:|\mathcal{S}^i|}$ will appear in the recommended list.

Here, following common symbolic notation, upper case bold letters denote matrices, lower case bold letters denote column vectors without any specification, and non-bold letters represent scalars. The major symbols are listed in Table 1.

Table 1: List of notations.

M, N	the number of users and items
$\mathcal{S}^i$	the item sequence of user i
$\mathbf{S}_{i,l}$	the embeddings of the l-th subsequence of user i
$\mathbf{W}_{g_*}, \mathbf{w}_{g_*}$	the learnable parameters in the gating layers
$\mathbf{U}$	the user embedding matrix
$\mathbf{E}$	the input item embedding matrix
$\mathbf{Q}$	the output item embedding matrix
d	the dimension of the embeddings
$\hat{r}_{i,j}$	the prediction score of user i on item j
λ	the regularization term

4 METHODOLOGIES

To model the sequential recommendation task, for each user i, we extract every $|L|$, i.e. $L = (\mathcal{S}^i_j, \mathcal{S}^i_{j+1}, ..., \mathcal{S}^i_{j+|L|-1})$, successive items as input and their next $|T|$ items as the targets to be predicted. The problem can be formulated as: in the user-item interaction sequence $\mathcal{S}^i$, given the $|L|$ successive items, how likely other N items will be interacted subsequently.

In the sequential recommendation problem, the prediction of users' preferences on items can be modeled in two perspectives: long-term interests and short-term interests. The long-term user preference modeling has been widely investigated in the conventional Top-N recommendation methods, such as matrix factorization [15, 29]. On the other hand, how to capture the short-term user interests from the sequential data is the key point for performance improvement.

For the short-term interest modeling, we argue that there are two kinds of relationships existing between items users have interacted and items users will interact in the future: group-level and instance-level relations. The group-level influence illustrates a phenomenon that several items in L together have an impact on the items user may interact in the future. For example, if a user has bought a bed frame and a mattress, a pillow is probably a more suitable recommendation than a table. On the other hand, the instance-level influence depicts the strong relation between a single item in L and a single item in T. For example, if a user bought a mobile phone, she may also need to buy a screen protector or a case. Thus, these two kinds of relations together determine users' short-term interests.

In this section, we introduce the proposed model to capture both the long-term interests and short-term interests of users for the sequential recommendation, which is shown in Figure 1 and Figure 2. We first illustrate the hierarchical gating network for learning users' group-level preferences. Next, we present the inner product of item embeddings to model the item-item relations. Then we introduce the prediction layer for aggregating the long-term and short-term interests of users. Lastly, we go through the loss function and training process of the proposed model.

4.1 Hierarchical Gating for Group-level Influence

In the sequential recommendation, taking advantage of the properties of sequential data to learn the (sub)sequence representation is a critical point, where an item may be closely related to its previous or subsequent items, or a group of previous items will have an impact on the items in the near future. In previous works, researchers have utilized various methods to model the group-level sequential interactions, e.g., convolutional neural networks [34], recurrent neural networks [12, 13, 27, 28], and the self-attention model [18]. Different from previous works, we propose a hierarchical gating network for modeling group-level user-item interactions, which consists of two components: a feature gating module and an instance gating module. These two modules allow the selection of effective latent features and relevant items, respectively, for predicting the subsequent items. Our proposed gating network is both effective and efficient (section 5).

4.1.1 *Feature Gating*

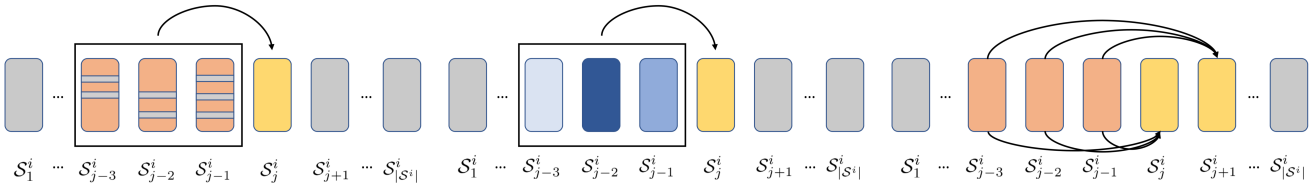
Unlike previous works [12, 18, 34] that only operate on the item-level, we provide a learnable feature gating module to select salient latent features of items from the feature-level. For a certain item, some parts of the latent features are more relevant to predict the subsequent items. For example, for a big fan of Robert Downey Jr., after watching Iron Man I and Iron Man II, it is better to recommend Iron Man III rather than Aquaman, although Aquaman is also a superhero movie. Thus, to capture the representative item features based on users' long-term preferences is a necessary point to capture.

Embedding Layer. In the proposed module, the input is a sequence of $|L|$ items, where each item is represented by a unique index. At the embedding layer, the item index is converted into a low-dimensional real-valued dense vector representation by an item embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times N}$, where d is the dimension of the item embedding and N is the number of items. After converted by the embedding layer, the item subsequence embeddings are represented as:

$$\mathbf{S}_{i,l} = \begin{bmatrix} & | & | & | & \\ ... & \mathbf{e}_{j-1} & \mathbf{e}_j & \mathbf{e}_{j+1} & ... \\ & | & | & | & \end{bmatrix}$$

where $\mathbf{S}_{i,l} \in \mathbb{R}^{d \times |L|}$ indicates the embeddings of the l-th subsequence of user i, $\mathbf{e}_j \in \mathbb{R}^d$ is the j-th column of the embedding matrix $\mathbf{E}$.

Gated Linear Unit. Inspired by the gated linear unit (GLU) proposed by Dauphin et al. in [4], which is utilized to control what information should be propagated for predicting the next word in the language modeling task, we also adopt a similar model to select



(a) Feature gating (b) Instance gating (c) Item-item product

Figure 1: An illustrative example of the feature gating, instance gating, and item-item product modules. In Figure 1a, the gray lines on items denote those latent features are masked off. In Figure 1b, the darker blue means the item is more important. In Figure 1c, the line linked between two items denotes the inner product, which captures the relations between the items users have accessed and the items users will access in the future.

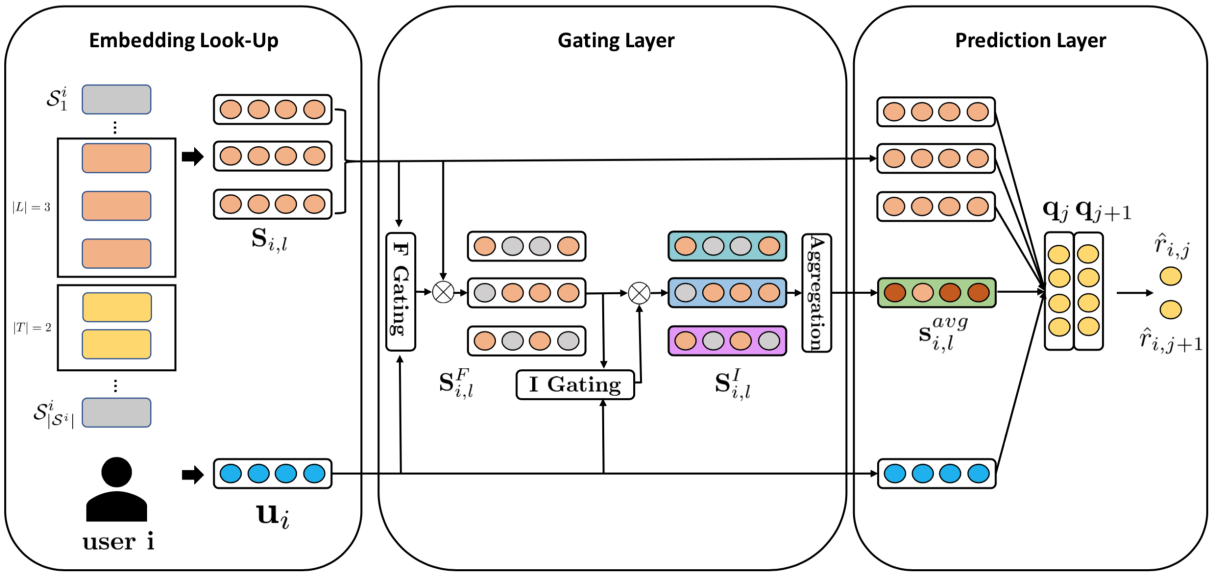


Figure 2: The architecture of HGN. HGN consists of three major components: the embedding layer, the hierarchical gating layer, and the prediction layer. Specifically, *F Gating* denotes the feature gating module, *I Gating* denotes the instance gating module, *Aggregation* denotes the aggregation layer, and ⊗ denotes the element-wise multiplication.

what features are relevant to predict future items. The GLU in the original paper is shown:

$$(\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c}),$$

where $\mathbf{X}$ is the input embeddings, $\mathbf{W}$, $\mathbf{V}$, $\mathbf{b}$, $\mathbf{c}$ are learnable parameters, σ is the sigmoid function, $*$ is the convolution operation, and $\otimes$ is the element-wise product between matrices.

Personalized Feature Gating. However, directly applying the GLU to select item features does not explicitly consider the user preference on items. For a certain item, a user may just focus a specific part of the item and neglect other unattractive parts. For example, a user may only care about whether the starring role is Tom Cruise rather than the movie content.

Therefore, to capture the item features that tailored to users' preferences, we need to modify the GLU to be user-specific. To reduce the number of learnable parameters, we apply the inner product instead of the convolution operation in the original GLU (the superscript F indicates the item sequence embeddings are learned from the feature gating module):

$$\mathbf{S}^F_{i,l} = \mathbf{S}_{i,l} \otimes \sigma(\mathbf{W}_{g_1} \cdot \mathbf{S}_{i,l} + \mathbf{W}_{g_2} \cdot \mathbf{u}_i + \mathbf{b}_g), \tag{1}$$

where $\mathbf{u}_i \in \mathbb{R}^d$ is the embedding of user i, $\mathbf{W}_{g_1}, \mathbf{W}_{g_2} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_g \in \mathbb{R}^d$ are learnable parameters, and $\otimes$ is the element-wise product between matrices. By doing this, user-specific features of items can be passed to downstream layers.

4.1.2 *Instance Gating*

Personalized Instance Gating. Since our formulated problem is: given $|L|$ successive items, how likely other items will appear after L in the near feature, we argue that there are some items are more relevant in L to predict the items users will interact. However, existing works either do not consider the representative items in L [12, 34] or apply attention models to capture the representative items [18, 20]. Unlike previous works benefiting from attention models, we adopt an instance-level gating module to select the informative items that are helpful to predict items in the near future

according to users' preferences:

$$\mathbf{S}^I_{i,l} = \mathbf{S}^F_{i,l} \otimes \sigma(\mathbf{w}^\top_{g_3} \cdot \mathbf{S}^F_{i,l} + \mathbf{u}^\top_i \cdot \mathbf{W}_{g_4}), \tag{2}$$

where $\mathbf{S}^I_{i,l} \in \mathbb{R}^{d\times|L|}$ is the sequence embedding after the instance gating, $\mathbf{w}_{g_3} \in \mathbb{R}^d$, $\mathbf{W}_{g_4} \in \mathbb{R}^{d\times|L|}$ are learnable parameters. By applying the instance gating, the representative items will contribute more to make predictions about the future items and irrelevant items will be largely neglected.

Aggregation Layer. To make the item embeddings $\mathbf{S}^I_{i,l}$ into one group-level latent representation, we can either apply average pooling or max pooling on $\mathbf{S}^I_{i,l}$:

$$\mathbf{s}^{avg}_{i,l} = avg - pooling(\mathbf{S}^I_{i,l}), \tag{3}$$

$$\mathbf{s}^{max}_{i,l} = max - pooling(\mathbf{S}^I_{i,l}), \tag{4}$$

where $\mathbf{s}^{avg}_{i,l}, \mathbf{s}^{max}_{i,l} \in \mathbb{R}^d$. Since the item embeddings have manipulated by the feature-level and instance-level gating modules, the informative features and items have been selected and irrelevant ones have been eliminated. Thus, the average pooling will accumulate the informative parts in these embeddings. On the other hand, max-pooling directly selects the most representative features from each embedding to form the group-level representation.

4.2 Item-item Product

The relation between two single items is an important factor to model in the recommendation task and has been widely studied in many years [17, 25], e.g., item-based collaborative filtering methods utilizing the rating vectors of two items to calculate the similarity. However, most of the recent works [12, 18, 34] only consider the sequential recommendation from the group-level, but do not explicitly capture the item-item relations between the items in L and the items user will interact in the future. Since strongly related item pairs will appear in L and T simultaneously. Unlike previous works, we apply the inner product between the input item embeddings and the output item embeddings to capture the item relations between L and T:

$$\sum_{\mathbf{e}_j \in \mathbf{S}_{i,l}} \mathbf{e}^\top_j \cdot \mathbf{Q},$$

where $\mathbf{Q} \in \mathbb{R}^{d\times N}$ is the output item embeddings, the sum of multiplication results captures the accumulated item-item relation scores from each item in L to all other items.

4.3 Prediction Layer

After applying the hierarchical gating network to capture the short-term interests of users and item-item product to capture the relevant item pairs, we adopt the classical matrix factorization term to capture the global and long-time interests of users. Given the l-th subsequence to predict, the prediction score of user i on item j is:

$$\hat{r}_{i,j} = \mathbf{u}^\top_i \cdot \mathbf{q}_j + \mathbf{s}^{avg\top}_{i,l} \cdot \mathbf{q}_j + \sum_{\mathbf{e}_k \in \mathbf{S}_{i,l}} \mathbf{e}^\top_k \cdot \mathbf{q}_j, \tag{5}$$

where $\mathbf{q}_j \in \mathbb{R}^d$ is the j-th column of the output item embedding $\mathbf{Q}$. In the prediction layer, the first term captures the user long-term interests, the second term models the user short-term interests, and the third term reflects the relations between item pairs.

4.4 Network Training

As the training data is from the user implicit feedback, we optimize the proposed model by the Bayesian Personalized Ranking objective [29]: optimizing the pairwise ranking between the positive and non-observing items:

$$\arg\min_{\mathbf{U},\mathbf{Q},\mathbf{E},\mathbf{\Theta}} \sum_{(i,L_i,j,k)\in\mathcal{D}} -log\sigma(\hat{r}_{i,j} - \hat{r}_{i,k}) + \lambda(||\mathbf{U}||^2 + ||\mathbf{Q}||^2 + ||\mathbf{E}||^2 + ||\mathbf{\Theta}||^2), \tag{6}$$

where L_i denotes one of the $|L|$ successive items of user i, j denotes the item that in T_i, and k denotes the randomly sampled negative item, $\mathbf{\Theta}$ is the parameters in the gating network, λ is the regularization parameter. By minimizing the objective function, the partial derivatives with respect to all the parameters can be computed by gradient descent with back-propagation. We apply Adam [19] to automatically adapt the learning rate during the learning procedure.

Time complexity. The computational complexity of our model for each L is mainly due to the feature gating layer and item-item product module, which is $O(|L|d^2 + |L|Nd)$ ($|L|$ is the length of L, d is the dimension of embeddings, and N is the number of items). This computational complexity makes our model scalable on large datasets. We empirically test the training speed with other state-of-the-art methods and find that our model is faster than other methods (section 5.7).

5 EXPERIMENTS

In this section, we evaluate the proposed model with the state-of-the-art methods on five real-world datasets[1].

5.1 Datasets

The proposed model is evaluated on five real-world datasets from various domains with different sparsities: *MovieLens-20M* [6], *Amazon-Books* and *Amazon-CDs* [9], *Goodreads-Children* and *Goodreads-Comics* [38]. *MovieLens-20M* is a user-movie dataset collected from the *MovieLens* website, where this dataset has 20 million user-movie interactions. The *Amazon-Books* and *Amazon-CDs* datasets are adopted from the Amazon review dataset[2] with different categories, i.e., CDs and Books, which cover a large amount of user-item interaction data, e.g., user ratings and reviews. *Goodreads-Children* and *Goodreads-Comics* datasets[3] are collected in late 2017 from *goodreads* website with different genres, and we use the genres of Children and Comics. In order to be consistent with the implicit feedback setting, we keep those with ratings no less than four (out of five) as positive feedback and treat all other ratings as missing entries on all datasets. To filter noisy data, we only keep the users with at least ten ratings and the items at least with five ratings. The data statistics after preprocessing are shown in Table 2.

For each user, we hold the 70% of interactions in the user sequence as the training set and use the next 10% of interactions as the validation set for hyper-parameter tuning. The remaining 20% constitutes the test set for reporting model performance. Note that during the testing procedure, the input sequences include the interactions in both the training set and validation set. The execution

[1]The code is available on Github: https://github.com/allenjack/HGN
[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home

of all the models is carried out five times independently, and we report the average results.

Table 2: The statistics of datasets.

Dataset	#Users	#Items	#Interactions	Density
ML20M	129,797	13,649	9,921,393	0.560%
Books	52,406	41,264	1,856,747	0.086%
CDs	17,052	35,118	472,265	0.079%
Children	48,296	32,871	2,784,423	0.175%
Comics	34,445	33,121	2,411,314	0.211%

5.2 Evaluation Metrics

We evaluate our model versus other methods in terms of *Recall@k* and *NDCG@k*. For each user, Recall@k (R@k) indicates what percentage of her rated items can emerge in the top k recommended items. NDCG@k (N@k) is the normalized discounted cumulative gain at k, which takes the position of correctly recommended items into account.

5.3 Methods Studied

To demonstrate the effectiveness of our model, we compare to the following recommendation methods.

Classical methods for implicit feedback:

- **BPRMF**, the Bayesian Personalized Ranking based matrix factorization [29], which is a classic method for learning pairwise personalized rankings from user implicit feedback. Specifically, we use BPR-MF for model learning.

State-of-the-art session-based recommendation methods:

- **GRU4Rec**, gated recurrent unit for recommendation [13], which uses recurrent neural networks to model user-item interaction sequences for session-based recommendation. Each user sequence is treated as a session.
- **GRU4Rec+**, an improved version of GRU4Rec [12], which adopts a different loss function and sampling strategy, and shows significant performance gains on Top-N recommendation.
- **NextItNet**, the next item recommendation net [42], applies dilated convolutional neural networks to increase the receptive fields without relying on the pooling operation.

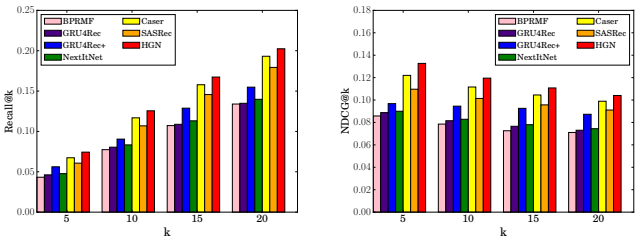
State-of-the-art sequential recommendation methods:

- **Caser**, convolutional sequence embedding model [34], which captures high-order Markov chains by applying convolution operations on the embeddings of the $|L|$ recent items.
- **SASRec**, self-attention based sequential model [18], which uses an attention mechanism to identify relevant items for predicting the next item.

The proposed method:

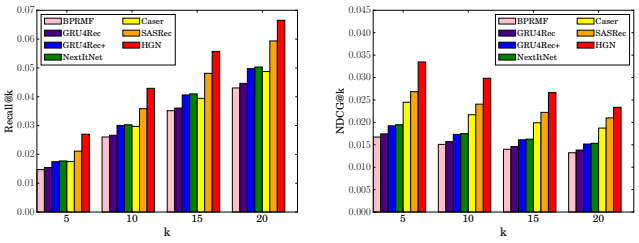
- **HGN**, the proposed model, applies a hierarchical gating network to learn the group-level representations of a sequence of items and adopts the item-item product to explicitly capture the item-item relations.

Given our extensive comparisons against the state-of-the-art methods, we omit comparisons with methods such as FMC and FPMC [30], Fossil [8], since they have been outperformed by the recently proposed Caser and SASRec.



(a) Recall@k on MovieLens-20M (b) NDCG@k on MovieLens-20M

Figure 3: The performance comparison on MovieLens-20M.



(a) Recall@k on Amazon-Books (b) NDCG@k on Amazon-Books

Figure 4: The performance comparison on Amazon-Books.

5.4 Experiment Settings

In the experiments, the latent dimension of all the models is set to 50. For those session-based methods, we treat each user sequence as one session. For GRU4Rec and GRU4Rec+, we find that when the learning rate is 0.001, and batch size is 50 can achieve good performance. These two methods adopt Top1 loss and BPR-max loss, respectively. For NextItNet, we following the original settings in the paper to set the learning rate to 0.001, the kernel size to 3, the dilated levels to 1 and 2, the batch size to 32. For Caser, we follow the settings in the author-provided code to set $|L| = 5$, $|T| = 3$, the number of horizontal filters to 16, the number of vertical filters to 4, where Caser can achieve good results. For SASRec, we set the number of self-attention blocks to 2, the batch size to 128, and the maximum sequence length to 50. The network architectures of above methods are also set the same with the original papers. The hyper-parameters are tuned using the validation set.

For HGN, we follow the same setting in Caser to set $|L| = 5$ and $|T| = 3$, where the length effects are shown in the section 5.8. Hyper-parameters are tuned by grid search on the validation set. The network embedding size d is also set to 50. The learning learning rate and λ are set to 0.001 and 0.001, respectively. The batch size is set to 4096. Our experiments are conducted with PyTorch[4] running on GPU machines (Nvidia GeForce GTX 1080 Ti).

5.5 Performance Comparison

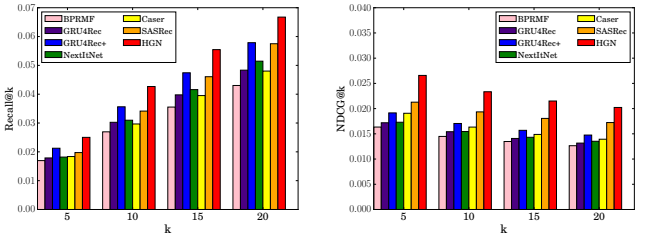
The performance comparison results are shown in Figure 3, 4, 5, 6, 7, and Table 3.

Observations about our model. First, the proposed model—HGN, achieves the best performance on five datasets with all evaluation metrics, which illustrates the superiority of our model. Second, HGN achieves better performance than SASRec. The reasons are three-fold: (1) SASRec only applies the instance-level selection but

[4]https://pytorch.org/

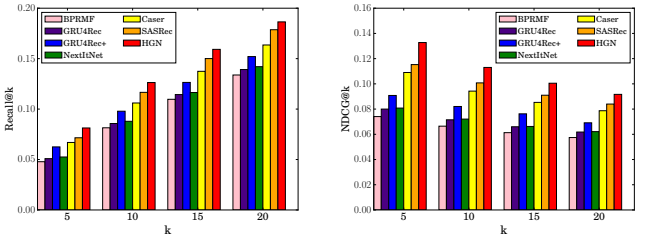
Table 3: The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method.

	BPRMF	GRU4Rec	GRU4Rec+	NextItRec	Caser	SASRec	HGN	Improv.
Recall@10								
MovieLens-20M	0.0774	0.0804	0.0904	0.0833	0.1169	0.1069	**0.1255***	7.36%
Amazon-Books	0.0260	0.0266	0.0301	0.0303	0.0297	0.0358	**0.0429*****	19.83%
Amazon-CDs	0.0269	0.0302	0.0356	0.0310	0.0297	0.0341	**0.0426****	19.66%
GoodReads-Children	0.0814	0.0857	0.0978	0.0879	0.1060	0.1165	**0.1263***	8.41%
GooReads-Comics	0.0788	0.0958	0.1288	0.1078	0.1473	0.1494	**0.1743*****	16.67%
NDCG@10								
MovieLens-20M	0.0785	0.0815	0.0946	0.0828	0.1116	0.1014	**0.1195***	7.07%
Amazon-Books	0.0151	0.0157	0.0173	0.0174	0.0216	0.0240	**0.0298*****	24.17%
Amazon-CDs	0.0145	0.0154	0.0171	0.0155	0.0163	0.0193	**0.0233****	20.73%
GoodReads-Children	0.0664	0.0715	0.0821	0.0720	0.0943	0.1007	**0.1130***	12.21%
GoodReads-Comics	0.0713	0.0912	0.1328	0.1171	0.1629	0.1592	**0.1927*****	18.29%



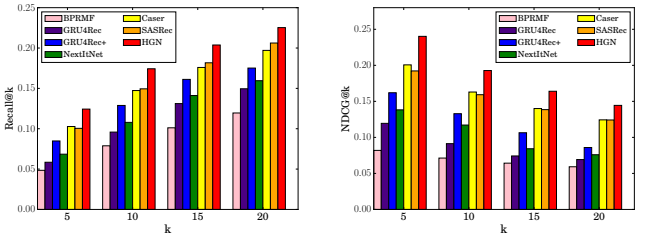
(a) Recall@k on Amazon-CDs **(b) NDCG@k on Amazon-CDs**

Figure 5: The performance comparison on Amazon-CDs.



(a) Recall@k on Children **(b) NDCG@k on Children**

Figure 6: The performance comparison on Children.



(a) Recall@k on Comics **(b) NDCG@k on Comics**

Figure 7: The performance comparison on Comics.

neglecting the feature-level one, which plays an important role in learning short-term user interests (section 5.6); (2) SASRec adopts a hyper-parameter—the maximum sequence length to reduce the computation burden, where only using part of the user data may lead to the insufficient understanding of long-term user interests; (3) SASRec does not explicitly model the item-item relations between two closely relevant items, which is captured by our item-item product module. Third, HGN outperforms Caser, one major reason is that Caser only applies CNNs to learn the group-level representation of several successive items without considering the item importance for different users. Fourth, HGN obtains better results than GRU4Rec, GRU4Rec+, and NextItNet. Two possible reasons are: (1) these models are session-based methods without explicitly modeling the long-term user interests; (2) these methods equally treat all the items in a short context, which may fail to capture the short-term user intentions. Fifth, HGN outperforms BPRMF. Since BPRMF only captures the long-term interests of users, which does not incorporate the sequential patterns of user-item interactions. On the top of BPRMF, HGN adopts a hierarchical gating network to capture the sequential dynamics in the user actions and an item-item product module to explicitly capture the item-item relations, which leads to better performance.

Other observations. First, all the results reported on MovieLens-20M, GoodReads-Children and GoodReads-Comics are better than the results on other datasets, the major reason is that other datasets are more sparse and the data sparsity declines the recommendation performance. Second, SASRec outperforms Caser on most of the datasets. The main reason is that SASRec adaptively attends items that would reflect the short-term user interests. Third, SASRec and Caser achieve better performance than GRU4Rec, GRU4Rec+, and NextItNet in most cases. One possible reason is that SASRec and Caser both explicitly plug the user embeddings in their models, which allows the long-term user interests modeling. Fourth, GRU4Rec+ performs better than other methods on one dataset. The reason is that GRU4Rec+ not only captures the sequential patterns in the user-item sequence but also has a promising object function—*BPR-max*. Fifth, all the methods perform better than BPR. This illustrates that only effectively modeling the long-term user interests is not sufficient to capture the user sequential behaviors.

5.6 Ablation Analysis

To verify the effectiveness of the proposed feature gating, instance gating, and item-item product modules, we conduct an ablation

Table 4: The ablation analysis on GoodReads-Comics and Amazon-Books datasets. *F* denotes the feature gating module, *I* denotes the instance gating module, *avg* denotes the average pooling, and *max* denotes the max pooling.

Architecture	*Comics*		*Books*	
	R@10	N@10	R@10	N@10
(1) BPR	0.0911	0.0802	0.0310	0.0177
(2) BPR+F+avg	0.1555	0.1624	0.0361	0.0266
(3) BPR+F+max	0.1456	0.1550	0.0355	0.0240
(4) BPR+I+avg	0.1538	0.1591	0.0351	0.0254
(5) BPR+I+max	0.1489	0.1585	0.0329	0.0241
(6) BPR+GRU	0.1456	0.1581	0.0289	0.0216
(7) BPR+CNN	0.1305	0.1387	0.0278	0.0207
(8) BPR+F+I+avg	0.1635	0.1791	0.0391	0.0250
(9) BPR+F+I+max	0.1569	0.1658	0.0355	0.0234
(10) HGN	**0.1743**	**0.1927**	**0.0429**	**0.0298**

analysis in Table 4 to demonstrate the importance each module contributes to the HGN model. In (1), we utilize only the BPR matrix factorization without any other components. In (2), we only incorporate the feature gating and apply the average pooling on the embeddings after the feature gating, on the top of (1). In (3), we replace the average pooling in (2) with max pooling. In (4), we only include the instance gating and apply the average pooling on the top of (1). In (5), we replace the average pooling in (4) with max-pooling. In (6), we adopt a recurrent neural network structure—gated recurrent unit (GRU) [3] to learn the group-level representations of items. In (7), we replace the GRU in (6) with a convolutional neural network (CNN), where the structure and hyper-parameters are set the same in Caser [34]. In (8), we both apply the feature and instance gating with average pooling. In (9), we replace the average pooling with max pooling. In (10), we present the overall HGN model to show the significance of the item-item product module.

From the results shown in Table 4, we have some observations. First, from (1) and all others, we can observe that the conventional BPR matrix factorization to capture the long-term user interests cannot effectively model the short-term user interests. Second, from (2), (3), (4) and (5), the feature gating seems to achieve slightly better results than the instance gating. And the average pooling is slightly better than the max pooling, one possible reason is that the average pooling makes the representative item features accumulated, which results in a more effective representation of a group of $|L|$ successive items. Third, from (6), (7), and (8), we observe that our hierarchical gating network achieves better performance than GRU and CNN but with *fewer* learnable parameters[5] (if we set the item embedding size to 50 ($d = 50$), then the number of learnable parameters of our hierarchical gating network is 5,350, the number of parameters of the one-recurrent-layer GRU is 15,300, the number of parameters of the CNN in [34] is 26,154). This result demonstrates that the proposed hierarchical gating network can effectively capture the sequential patterns in the user-item interaction sequence. Lastly, from (1), (8), and (9), we observe that by incorporating the item-item product, the performance further improves. The results demonstrate that explicitly capturing the relations between the items users accessed and those items users may interact in the future can provide a significant supplementary to model the user sequential dynamics.

[5]We verified the number of parameters of all three models by the *named_parameters()* function provided by PyTorch.

5.7 Training Efficiency

In this section, we evaluate the training efficiency with other state-of-the-art methods in terms of the training speed (time taken for one epoch of training). Since GRU4Rec+ has been compared with SASRec in [18], we omit the training time comparison with GRU4Rec+. To make a fair comparison, we set the max sequence length of SASRec as 300 to cover more than 95% of the sequence. All the experiments are conducted on a single GPU of Nvidia GeForce GTX 1080 Ti. All the compared methods are executed 20 epochs and we report the average computation time, which is shown in Table 5. Note that the time reported only includes the training time of models without including the negative sampling time.

Table 5: The training time per epoch comparison on five datasets in terms of seconds.

	CDs	Books	ML20M	Children	Comics
HGN	**0.957s**	**2.086s**	**28.304s**	**3.496s**	**2.228s**
SASRec	2.242s	16.154s	39.937s	14.913s	10.468s
Caser	5.063s	17.577s	63.702s	28.593s	25.657s

From the results in Table 5, we can observe that HGN yields the fastest training speed on all datasets. As we have discussed in section 4.4, our model has less item complexity than SASRec, which is $O(N^2d + Nd^2)$. Thus, our proposed model has better training efficiency both theoretically and practically.

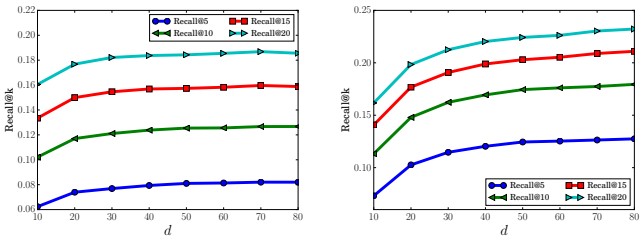
Table 6: The effect of the length $|L|$ and $|T|$.

Settings	*CDs*		*Comics*	
	R@5	R@10	R@5	R@10
$\|L\|$=3, $\|T\|$=1	0.0260	0.0415	0.1202	0.1684
$\|L\|$=3, $\|T\|$=2	**0.0291**	0.0448	0.1275	0.1758
$\|L\|$=3, $\|T\|$=3	0.0289	0.0450	**0.1296**	**0.1793**
$\|L\|$=5, $\|T\|$=1	0.0254	0.0417	0.1155	0.1645
$\|L\|$=5, $\|T\|$=2	0.0261	0.0432	0.1215	0.1711
$\|L\|$=5, $\|T\|$=3	0.0290	**0.0456**	0.1238	0.1738
$\|L\|$=8, $\|T\|$=1	0.0220	0.0372	0.1083	0.1566
$\|L\|$=8, $\|T\|$=2	0.0248	0.0401	0.1142	0.1636
$\|L\|$=8, $\|T\|$=3	0.0260	0.0413	0.1160	0.1658

5.8 The Sensitivity of Hyper-parameters

We present the effect of two hyper-parameters: the dimension of the item embeddings d and the length of successive items $|L|$ and $|T|$. The effects of these two parameters are shown in Figure 8 and Table 6. Due to the space limit, we only present the effects on two datasets, the parameter effects on other datasets have similar trends.

The variation of d is shown in Figure 8. We can observe that a small dimension of item embeddings is not sufficient to express the latent features of items. By increasing the dimension of item embeddings, the model has more capacity to model the complex features of items. With the increase of d, the model performance largely improves and becomes steady.



(a) d on Children (b) d on Comics

Figure 8: The dimension variations of embeddings.

The variation of $|L|$ and $|T|$ is shown in Table 6. We observe that when $|L|$ is fixed, a larger value of $|T|$, i.e. 3, can achieve better performance. This may illustrate that a group of $|L|$ items may determine several items that user will interact in the near future. We also observe that smaller $|L|$ has better results than larger ones. One possible reason is that larger $|L|$ may include too many irrelevant items for predicting future items.

6 CONCLUSION

In this paper, we propose a hierarchical gating network with an item-item product module for the sequential recommendation. The model adopts a feature gating module and an instance gating module to control what item features can be passed to downstream layers, where informative latent features and items can be selected. Moreover, we apply an item-item product module to capture the relations between closely relevant items. Experimental results on five real-world datasets clearly validate the performance of our model over many state-of-the-art methods and demonstrate the effectiveness of the gating and item-item product modules.

REFERENCES

[1] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential Recommendation with User Memory Networks. In *WSDM*. ACM, 108–116.

[2] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *IJCAI*. IJCAI/AAAI, 2605–2611.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. ACL, 1724–1734.

[4] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 933–941.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *IJCAI*. ijcai.org, 1725–1731.

[6] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 19:1–19:19.

[7] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *RecSys*. ACM, 161–169.

[8] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *ICDM*. IEEE, 191–200.

[9] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. ACM, 507–517.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. ACM, 173–182.

[11] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR*. ACM, 549–558.

[12] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *CIKM*. ACM, 843–852.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015).

[14] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *WWW*. ACM, 193–201.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. IEEE Computer Society, 263–272.

[16] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. ACM, 505–514.

[17] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *KDD*. ACM, 659–667.

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. IEEE Computer Society, 197–206.

[19] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[20] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. ACM, 1419–1428.

[21] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *KDD*. ACM, 1754–1763.

[22] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. 2016. Modeling User Exposure in Recommendation. In *WWW*. ACM, 951–961.

[23] Chen Ma, Peng Kang, Bin Wu, Qinglong Wang, and Xue Liu. 2019. Gated Attentive-Autoencoder for Content-Aware Recommendation. In *WSDM*. ACM, 519–527.

[24] Chen Ma, Yingxue Zhang, Qinglong Wang, and Xue Liu. 2018. Point-of-Interest Recommendation: Exploiting Self-Attentive Autoencoders with Neighbor-Aware Influence. In *CIKM*. ACM, 697–706.

[25] Xia Ning, Christian Desrosiers, and George Karypis. 2015. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In *Recommender Systems Handbook*. Springer, 37–76.

[26] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In *ICDM*. IEEE Computer Society, 502–511.

[27] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David M. J. Tax. 2017. Interacting Attention-gated Recurrent Networks for Recommendation. In *CIKM*. ACM, 1459–1468.

[28] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*. ACM, 130–137.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. AUAI Press, 452–461.

[30] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.

[31] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML (ACM International Conference Proceeding Series)*, Vol. 227. ACM, 791–798.

[32] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. ACM, 285–295.

[33] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NIPS*. 2440–2448.

[34] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. ACM, 565–573.

[35] Thanh Tran, Kyumin Lee, Yiming Liao, and Dongwon Lee. 2018. Regularizing Matrix Factorization with User and Item Embeddings for Recommendation. In *CIKM*. ACM, 687–696.

[36] Thanh Tran, Xinyue Liu, Kyumin Lee, and Xiangnan Kong. [n.d.]. Signed Distance-based Deep Memory Recommender.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 6000–6010.

[38] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *RecSys*. ACM, 86–94.

[39] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *KDD*. ACM, 1235–1244.

[40] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. ACM, 153–162.

[41] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. ijcai.org, 3203–3209.

[42] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*. ACM, 582–590.