Recommender System Using Sequential and Global Preference via Attention Mechanism and Topic Modeling

Kyeongpil Kang
Korea University & SKT AI Center
Seoul, South Korea
rudvlf0413@korea.ac.kr

Junwoo Park
SK Planet
Seoul, South Korea
junwoospark@gmail.com

Wooyoung Kim
Korea University & SKT AI Center
Seoul, South Korea
ddsksw01@sk.com

Hojung Choe
NICE Information Service
Seoul, South Korea
topaz1996@naver.com

Jaegul Choo
Korea University
Seoul, South Korea
jchoo@korea.ac.kr

ABSTRACT

Deep neural networks improved the accuracy of sequential recommendation approach which takes into account the sequential patterns of user logs, e.g., a purchase history of a user. However, incorporating only the individual's recent logs may not be sufficient in properly reflecting global preferences and trends across all users and items. In response, we propose a self-attentive sequential recommender system with topic modeling-based category embedding as a novel approach to exploit global information in the process of sequential recommendation. Our self-attention module effectively leverages the sequential patterns from the user's recent history. In addition, our novel category embedding approach, which utilizes the information computed by topic modeling, efficiently captures global information that the user generally prefers. Furthermore, to provide diverse recommendations as well as to prevent overfitting, our model also incorporates a vector obtained by random sampling. Experimental studies show that our model outperforms state-of-the-art sequential recommendation models, and that category embedding effectively provides global preference information.

CCS CONCEPTS

• **Information systems** → *Collaborative filtering*; • **Computing methodologies** → *Topic modeling*.

KEYWORDS

Recommender system, Deep neural networks, Topic modeling

ACM Reference Format:
Kyeongpil Kang, Junwoo Park, Wooyoung Kim, Hojung Choe, and Jaegul Choo. 2019. Recommender System Using Sequential and Global Preference via Attention Mechanism and Topic Modeling. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/335734.3358054

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/335734.3358054

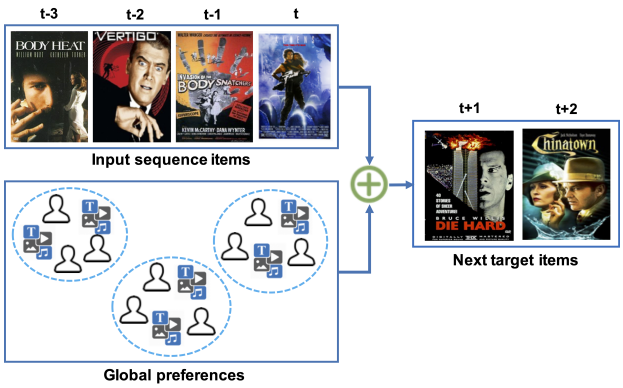


Figure 1: Recommender system using sequential and global preferences

1 INTRODUCTION

As the usage of online social media and e-commerce has grown dramatically, the online commercial contents increase at an exponential rate. Therefore, individuals are overwhelmed by the choices they encounter. To resolve this, recommendation approaches are employed across various fields, such as online market, and video streaming service, to guide those in need. For instance, the representative recommender system challenge was held by Netflix[1] to improve the performance of the recommender system. Since then, the recommender system assists users to select their preferred items among the vast contents in an efficient way.

We point out three important keys in recommendation approaches to improve the performance of recommendation approaches and apply to the real-world applications; 1) sequential item-item interactions that capture a short-term history of sequence within items, 2) global interaction such as user-category and user-user in the similar preference group, and 3) a diverse recommendation which recommends various items within the user's preference.

First of all, sequential item-item interaction focuses primarily on the user's short-term sequential pattern that originates from the previous choices. At first, Markov chain based methods were implemented with the transition matrix to predict the probability of the next item. However, with advancements in deep neural networks, many models adopt deep neural architectures such as

[1] https://www.netflixprize.com/

recurrent neural networks (RNNs) and convolutional neural networks (CNNs) to properly represent sequential information. On the other hand, global interaction focused on global information has built up throughout the entire histories from all users and items, extracting latent groups among users and items. Such interaction is dealt with conventional approaches including non-negative matrix factorization (NMF), latent Dirichlet allocation (LDA), and clustering methods. Moreover, the diverse recommendation has become important for real-world applications as the preference of each user varies depending on the situation. Therefore, we need to provide various recommendation options to match the user's dynamic preferences.

Integration of these factors, as shown in Fig. 1 plays a crucial role in the advance of the sequential recommender system as many users have both sequential and global preference. Besides, users interact with other users and categories in a similar preference group. We propose an approach to capture dynamic sequential patterns and exploit the global user-item interaction information, and a novel approach to successfully combine sequential and global information into a sequential recommender system framework. Our model outperforms other sequential recommendation models in both MovieLens and Gowalla datasets and also has the ability to deliver diverse recommendations to users.

Overall, our main contributions include the following:

(1) We propose a self-attentive sequential recommender system using the global structure as a novel approach to exploit both sequential and global information.
(2) We further propose a categorical preference gating to combine both sequential and global information. Our proposed gating mechanism effectively decides which information should be more utilized.
(3) In addition, we show that our model successfully provides users with diverse recommendations while preventing user information loss and overfitting.
(4) By applying our proposed method, our proposed model outperforms other state-of-the-art baseline models.

2 RELATED WORK

Traditional collaborative filtering mainly focused on obtaining relationships between users and items. Conventional recommendation approaches recommend items to the user by finding similar items that are not seen by users. Various algorithms such as clustering [8, 36], probabilistic graphical model [6, 23, 32], and matrix factorization [18, 20, 46] have been proposed in order to find similar items in the cluster or a latent space. Although such systems are being substituted by modern recommendation models such as deep neural networks due to lack of performance, they have the capability to regard global user-item interactions. Accordingly, to exploit global information, we adopt latent Dirichelet allocation (LDA) [3] that can learn global patterns from the user-item interactions.

Another reason behind disinterest in conventional collaborative filtering approaches is that it is difficult to represent sequential information. Sequential recommendation approaches specialize in foreseeing interactions within items generated by the user's sequential patterns. In addition, it is robust to the cold start problem since it considers only a few items in the sequence. Therefore, sequential recommendation approaches become more important. From the beginning, Markov chains were frequently used to exploit sequential patterns [4, 9, 10, 31, 43].

Later on, the paradigm shifted to applying deep neural architectures such as RNNSs [7, 12, 13, 27, 29, 40, 46] and CNNs [11, 34, 39] to represent sequential interactions. However, RNNs are vulnerable to the gradient vanishing and exploding problem, and CNNs have difficulties in figuring out the similarities between items that are far away from each other. In addition, they lack in obtaining global relationships within the entire users and items.

Recently, several approaches to using attention mechanisms [21, 25, 29, 33, 42, 44] were proposed to efficiently represent sequential interactions or relationship between user and item. In addition, self-attention mechanism [37] was proposed to efficiently represent sequential interactions by considering their own relationships. In other words, self-attention transforms sequential representation by matching a sequence with itself and then enabling attention to influence each other. By applying attention mechanism, self-attention not only represents sequential information efficiently but also achieves high performance through a high-level semantic combination of elements in the sequence. Several surveys adopted the self-attention to represent sequential information efficiently [14, 45].

Our proposed model employs self-attention to exploit item-item sequential interactions. Additionally, it uses LDA to further capture global information that the self-attention model may neglect. We integrate self-attention and LDA by applying our newly proposed preference gate. In addition, a diverse recommendation in order to provide various options to users is an important topic in the field of recommender system because users sometimes want to request recommendation service again when users unsatisfied the recommended options provided by the recommender system. Several studies add the random noise vector to the content vector in order to generate various images based on the content vector [15, 17]. We apply this technique to provide a diverse recommendation.

3 PROPOSED METHODS

We propose a novel sequential recommender system to provide efficient sequential representation. The proposed model, SelCa (Self-attentive categorical recommendation), incorporates self-attention to create an efficient sequential representation and LDA to exploit global information by providing entire interactions from all users and items. In addition, we add a random noise vector to the user embedding vector to deliver diverse recommendations to the users while avoiding performance loss.

As shown in Fig. 2, our model mainly consists of four main modules; (a) a self-attention module, (b) a category embedding module, (c) a user representation module, and (d) a categorical preference gate. Specifically, in order to combine categorical and sequential representation efficiently, we propose the categorical preference attention gate. In the gate, we apply element-wise multiplication input, which is widely used in natural language processing tasks [5, 26]. The reason for this input is to provide element-wise distance information for each dimension.

The training approach of our model is similar to Caser model [34]. Suppose that there exist a set $\mathcal{U}$ of M users and a set $\mathcal{I}$ of N

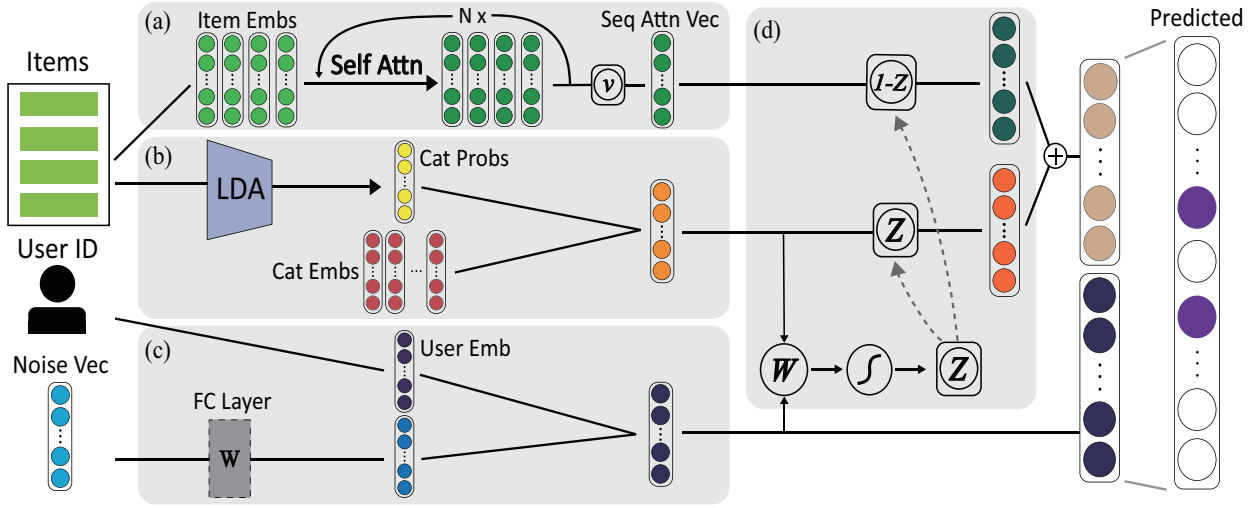


Figure 2: Overview of our proposed model, SelCa.

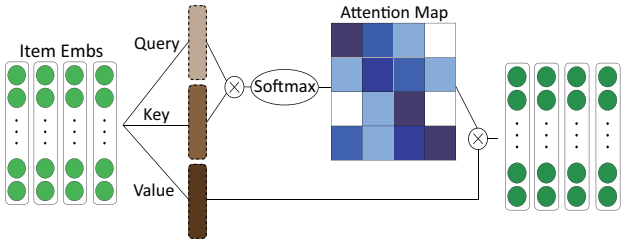


Figure 3: Our self-attention-based item encoding

items. For each user $u \in \mathcal{U}$ and his/her previous items $\mathcal{L}_{u,t} = \{i_{u,t-|\mathcal{L}|+1}, \cdots, i_{u,t}\}$, where each $i_{u,t} \in \mathcal{I}$ is an item watched by user u in the time period of t, our model predicts preference scores of the next target items $\mathcal{T}_{u,t} = \{i_{u,t+1}, \cdots, i_{u,t+|\mathcal{T}|}\}$. Our proposed model is optimized by maximizing the predicted preference scores of the next target items $\mathcal{T}_{u,t}$.

3.1 Item Encoding via Self-Attention Module

A recently proposed self-attention module has been widely used to properly represent sequential vector representations [37]. As shown in Fig. 3, the self-attention module efficiently transforms vector representations from a sequence of item vectors by considering their semantic relations. Self-attention calculates vector representations using the notion of a query, a key, and a value, i.e.,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\mathbf{Q}\mathbf{K}^\top\right)\mathbf{V}, \tag{1}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ represent a query, a key, and a value matrices, respectively.

A self-attention module, shown in Fig. 2-(a), obtains d-dimensional embedding vectors $\{\mathbf{e}_1^{item}, \cdots, \mathbf{e}_{|\mathcal{L}|}^{item}\}$ from item embedding matrix $\mathbf{E}^{item} \in \mathbb{R}^{d \times N}$ corresponding to the sequence of items $\mathcal{L}_{u,t}$. After packing item embedding vectors into $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, attention module transforms the embedding vectors to $\mathbf{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_{|\mathcal{L}|}\}$. Similar to the original self-attention model [37], we adopt additional techniques such as positional encoding, multi-head attention, scaled-dot product, and position-wise feed-forward networks. A transformation process is repeated n_{att} times with residual connection between each encoding process, i.e.,

$$\mathbf{S} = \mathbf{S} + \text{Attention}(\mathbf{S}). \tag{2}$$

Afterward, the self-attentive vector $\mathbf{v}_a$ is gained by combining $|\mathcal{L}|$ number of attention vectors, i.e.,

$$\mathbf{v}_a = \sum_{i=1}^{|\mathcal{L}|} \mathbf{v}_i^a \, \mathbf{s_i}, \tag{3}$$

where $\mathbf{v}^a \in \mathbb{R}^{|\mathcal{L}|}$ is a trainable weight vector to capture the importance of sequential position.

3.2 Category Representation

In order to exploit global information from all users and items, we use LDA [2] to calculate the probabilities of a sequence of items that belong to each category. First of all, we assume that there are k latent categories in the dataset. To train LDA, first we construct user-item matrix $\mathbf{D} \in \mathbb{R}^{M \times N}$, where $\mathbf{D}_{i,j}$ is the number of watching events of item j by user i. LDA is trained to maximize the marginal probability of each category of given sequence and items in the training dataset.

After training LDA, each probability $\mathbf{p}_l \in \mathbb{R}^k$ is calculated, where $p_{l|k}$ represents the probability of the given sequence l belongs to category k. Finally, as shown in Fig. 2-(b), categorical representation $\mathbf{v}_{cat}$ is computed using $\mathbf{p}_l$ and d-dimensional category embedding vectors $\{\mathbf{e}_1^{cat}, \cdots, \mathbf{e}_k^{cat}\} \in \mathbf{E}^{cat}$ as

$$\mathbf{v}_{cat} = \sum_{c=1}^{k} \mathbf{p}_{l,c} \, \mathbf{e}_c^{cat}, \tag{4}$$

where category embedding vectors $\mathbf{e}_c^{cat}$ are parameters that are updated during the training process.

[2] Although other methods such as NMF also can be used, we applied LDA because it yields topic probabilities that range from 0 to 1, which makes the training process stable.

3.3 User Embedding

For the characteristics of agents with variable preferences, many applications need diverse recommendations. To cover this functionality, we add the Gaussian noise vector $\mathbf{z} \sim \mathcal{N}(0,1) \in \mathbb{R}^d$ to the user embedding vector $\mathbf{e}_u^{user} \in \mathbb{R}^d$. However, if the noise vector is added to the user embedding vector directly, it is prone to loss user preference information, which causes the decrements of model performance. Therefore, as shown in Fig. 2-(c), we added the noise vector to the user embedding vector v_u after linearly transforming the noise vector as

$$\mathbf{v}_u = \mathbf{e}_u^{user} + Linear(\mathbf{z}). \tag{5}$$

Linear transformation to the random noise vector is applied in several researches [15] as it can preserve the model performance.

3.4 Categorical Preference Gate

After calculating the self-attentive vector $\mathbf{v}_a$, category embedding vector $\mathbf{v}_{cat}$, and user embedding vector $\mathbf{v}_u$, we calculate the categorical preference gate, as shown in Fig. 2-(d), in order to combine the category embedding vector and the self-attentive vector efficiently. The categorical preference gate is calculated as

$$g = \sigma(\mathbf{W}_g[\mathbf{v}_{cat}; \mathbf{v}_u; \mathbf{v}_{cat} \circ \mathbf{v}_u] + b_g), \tag{6}$$

where $\circ$ represents the Hadamard product or element-wise multiplication, and σ represents the sigmoid function to normalize the input value to the range [0, 1].

After calculating the gate score g, we obtain the content vector $\mathbf{v}_{content}$ by combining the self-attentive vector and the category embedding vector using the gate score g, i.e.,

$$\mathbf{v}_{content} = g * \mathbf{v}_a + (1-g) * \mathbf{v}_{cat}. \tag{7}$$

By applying this categorical preference gate instead of concatenating two vectors, we can combine the self-attentive vector v_a and the category embedding vector v_{cat} more efficiently by considering the user's preference while reducing the number of parameters.

3.5 Prediction Layer

Finally, after combining the content vector $\mathbf{v}_{content}$ and the user embedding vector $\mathbf{v}_u$, the preference score of item i is predicted as

$$p_i = \sigma(\mathbf{W}_i^\top [\mathbf{v}_{content}; \mathbf{v}_u] + \mathbf{b}_i), \tag{8}$$

where $\mathbf{W} \in \mathbb{R}^{N \times 2d}$ and $\mathbf{b} \in \mathbb{R}^N$ are parameters for last fully connected layer. For the inference step, we choose the top-ranked item from the predicted preference scores.

3.6 Optimization of Our Model

Since the task is to predict multiple items in the target sequence $\mathcal{T}_{u,t}$, this task can be viewed as multi-label classification. The model is trained to minimize the negative log-likelihood of target items for the given previous item sequence corresponding to each user. In other words, for the given previous item sequence $\mathcal{L}_{u,t}$ and the next target items $\mathcal{T}_{u,t}$ from the user u, the negative log-likelihood loss is calculated as

$$\mathcal{L}(u, \mathcal{L}_{u,t}, \mathcal{T}_{u,t}; \theta) = -\sum_{i \in \mathcal{T}_{u,t}} \sum_{j \in \mathcal{N}(u)} (\log p_i - \log p_j), \tag{9}$$

Dataset	MovieLens	Gowalla
#(Users)	6.0K	13.1K
#(Items)	3.4K	14.0K
#(Total actions)	1.10M	0.59M
#(Actions per user)	165.50	40.74
Sparsity	95.2%	99.7%
Sequential intensity	0.3265	0.0748

Table 1: Datasets statistics

where $\mathcal{N}(u)$ is the set of negative items randomly sampled from the user u. The model is optimized by minimizing the total loss, i.e.,

$$\min_\theta \sum_u \sum_t \mathcal{L}\left(u, \mathcal{L}_{u,t}, \mathcal{T}_{u,t}; \theta\right). \tag{10}$$

In addition, we pretrain Doc2Vec [19] before training the main recommendation model to initialize user embedding vectors $\mathbf{E}^{user}$ and item embedding vectors $\mathbf{E}^{item}$. Additionally, for initialization of category embedding vectors $\mathbf{E}^{cat}$, we first obtain the probability $p_{i,c}$, which is the probability of item i to be assigned to the c-th category, using LDA. After then, each category embedding vector $\mathbf{e}_c^{cat}$ is calculated as

$$\mathbf{e}_c^{cat} = \sum_i^N p_{i,c}\, \mathbf{e}_i^{item}. \tag{11}$$

Such pretraining of embedding vectors stabilizes the training process and accelerates the convergence speed of the training loss. On the other hand, Gowalla dataset, as shown in Table 1, has a small number of actions per user, thus preventing our Doc2Vec pretraining step from being properly trained. To resolve this, we instead apply random initialization for the Gowalla dataset. We find that the model with random initialization instead of Doc2Vec performs better in the MovieLens dataset.

4 EXPERIMENTS

In this section, we describe our experimental settings and baseline models.

4.1 Datasets

In order to train the sequential recommendation models, the sequential pattern should exist in the dataset. To distinguish the datasets with the sequential signal, Tang and Wang defines the sequential intensity (SI) and estimates the degree of patterns for each dataset. SI is the total number of association rules divided by the total number of users, where the association rule [1, 2] represents an implication $\mathbf{X} \Rightarrow \mathbf{y}$. $\mathbf{X}$ is a set of $|\mathcal{L}|$ items in $\mathcal{I}$, and $\mathbf{y}$ is an item in $\mathcal{I}$ that is not in $\mathbf{X}$. We select MovieLens[3] and Gowalla[4] as our datasets for the experiments as summarized in Table 1. We apply the association rule mining algorithm to the sequential actions of the entire users to find the sequential pattern in the same method as [34].

The MovieLens dataset contains movie ratings and is widely used to evaluate numerous recommendation models. Gowalla dataset provides location-based check-in information and is widely used in the point-of-interest (PoI) recommendation. In the same manner as [34], we converted the movie rating of the user into implicit

[3] https://grouplens.org/datasets/movielens/
[4] https://snap.stanford.edu/data/loc-gowalla.html

feedback of 1 and removed the users and the items which received the feedback less than n. n is set as 5 for MovieLens and 15 for Gowalla, respectively. We used the first 70% of each user-specific actions in each dataset for the training set, the next 10% for the validation set, and the remaining 20% for the test set.

4.2 Experimental Setup

We used the Adam optimizer [16] for optimizing the loss function in Eq. (10). For the MovieLens dataset, we set the learning rate to 0.001, the learning rate decay to 0.5 every 8 epochs, dimension d to 100, the number of categories k to 30, and the sequence length $|\mathcal{L}|$ to 9. For the Gowalla dataset, we set the learning rate to 0.01, the learning rate decay to 0.25 every 10 epochs, dimension d to 150, the number of categories k to 5, and the sequence length to $|\mathcal{L}|$. For both datasets, we set the weight of L_2-regularization to 10^{-6}, the target sequence length $|\mathcal{T}|$ to 3, the number of layers n_{att} in the self-attention module to 5, and the number of heads in the self-attention layer to 2. We implemented and trained our model using PyTorch Library and GPU TITAN Xp.

4.3 Evaluation Metrics

Precision, recall, and mean average precision (mAP) are used for the evaluation in the same manner as [28, 31, 34, 38, 41]. For the given top k-ranked predicted items $\hat{Y}_u$ from the user u and the last 20% of real items Y_u in test set, Prec@k and Recall@k are calculated as

$$Prec@k(u) = \frac{|Y_u \cap \hat{Y}_u|}{k}, \tag{12}$$

$$Recall@k(u) = \frac{|Y_u \cap \hat{Y}_u|}{|\hat{Y}_u|}. \tag{13}$$

Moreover, mAP scores are calculated by averaging all users' average precision scores as

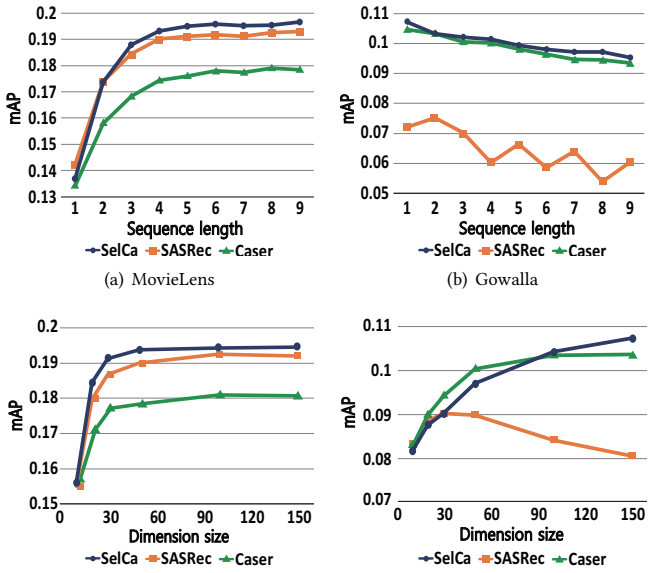
$$mAP = \frac{1}{M}\frac{1}{|\mathcal{K}|}\sum_{u\in\mathcal{U}}\sum_{k\in\mathcal{K}} Prec@k(u). \tag{14}$$

We conducted experiments with $\mathcal{K} = \{1, 5, 10\}$ for each precision and recall and $\mathcal{K} = \{1, 2, \cdots, |Y_u|\}$ for mAP score.

4.4 Baselines

Baseline models that we used for the model comparison are described as follows.

(1) **POP** : POP ranks the items by their popularity from all of the users' sequences. By using the number of interactions, POP recommends the most viewed items.
(2) **BPR** : The Bayesian personalized ranking model [30] factorizes the user-item interaction matrix using the BPR loss.
(3) **FMC**, **FPMC** : Factorized Markov chain (FMC) [31] factorizes the first-order Markov transition matrix. Furthermore, factorized personalized Markov chain (FPMC) [43] additionally applies the latent factor model.
(4) **GRU4Rec** : GRU4Rec [13] is the early model that adopts the RNN architecture to represent the sequential information.
(5) **Caser** : Caser [34] learns the sequential patterns and general user preferences using horizontal and vertical convolution filters. Caser is one of the best performing models among the sequential recommenders.



(a) MovieLens (b) Gowalla (c) MovieLens (d) Gowalla

Figure 4: mAP scores along the sequence length L and the dimension size d.

(6) **SASRec** : SASRec [14] adopts self-attention networks, and it is one of the best performing models among the sequential recommender models.

5 RESULTS

In this section, we describe experimental results with our model and baseline models, and we explain the effects of self-attention, category embeddings, and the categorical preference gate. In addition, we describe the results of our proposed diverse recommendation.

5.1 Prediction Accuracy

All the performance results of our model and baseline models are reported in Table 2. In this paper, we highly optimize hyperparameters in the Caser model and record better performance than the original paper [34]. We additionally reported confidence intervals for top-three models, and these confidence intervals show that our results are statistically significant. Our model outperforms all of the other baseline models. Generally, Performances using Gowalla dataset are lower than those of MovieLens dataset due to its sparsity as shown in Table 1. Interestingly, the performance of SASRec is much lower than the Caser and our model. This is because SASRec takes into account only sequential representation and therefore has the little capability in the sparse dataset. This also shows that our model is more efficient than SASRec model in sparse datasets as shown in Fig. 4. As the dimension size increases to 150, performances of our model and Caser increase, but SASRec does not.

5.2 Ablation Test

Further, we conduct ablation test using our model. After removing each module, we calculate the mAP score for each dataset using each model respectively.

Dataset	Metric	POP	BPR	FMC	FPMC	GRU4Rec	Caser	SASRec	SelCa	Margin
MovieLens	Prec@1	0.1280	0.1478	0.1748	0.2022	0.2515	0.3006 (± 0.0003)	<u>0.3261</u> (± 0.0015)	**0.3359** (± 0.0014)	3.0%
	Prec@5	0.1113	0.1288	0.1505	0.1659	0.2146	0.2655 (± 0.0005)	<u>0.2782</u> (± 0.0007)	**0.2859** (± 0.0005)	2.8%
	Prec@10	0.1011	0.1193	0.1317	0.1460	0.1916	0.2366 (± 0.0004)	<u>0.2471</u> (± 0.0005)	**0.2540** (± 0.005)	2.8%
	Recall@1	0.0050	0.0070	0.0104	0.0118	0.0153	0.0203 (± 0.0005)	<u>0.0216</u> (± 0.0002)	**0.0222** (± 0.0001)	2.8%
	Recall@5	0.0213	0.0312	0.0432	0.0468	0.0629	0.0835 (± 0.0002)	<u>0.0876</u> (± 0.0003)	**0.0894** (± 0.0002)	2.1%
	Recall@10	0.0375	0.0560	0.0722	0.0777	0.1093	0.1438 (± 0.0002)	<u>0.1501</u> (± 0.0004)	**0.1534** (± 0.0003)	2.2%
	mAP	0.0687	0.0913	0.0949	0.1053	0.1440	0.1801 (± 0.0003)	<u>0.1902</u> (± 0.0004)	**0.1967** (± 0.0002)	3.4%
Gowalla	Prec@1	0.0517	0.1640	0.1532	0.1555	0.1050	<u>0.2125</u> (± 0.0013)	0.1564 (± 0.0048)	**0.2225** (± 0.0007)	4.7%
	Prec@5	0.0362	0.0983	0.0876	0.0936	0.0721	<u>0.1220</u> (± 0.0004)	0.1004 (± 0.0021)	**0.1273** (± 0.0003)	4.3%
	Prec@10	0.0281	0.0726	0.0657	0.0698	0.0571	<u>0.0912</u> (± 0.0002)	0.0768 (± 0.0015)	**0.0951** (± 0.0002)	4.3%
	Recall@1	0.0064	0.0250	0.0234	0.0256	0.0155	<u>0.0345</u> (± 0.0002)	0.0242 (± 0.0009)	**0.0360** (± 0.0001)	4.4%
	Recall@5	0.0257	0.0743	0.0648	0.0722	0.0529	<u>0.0931</u> (± 0.0002)	0.0747 (± 0.0021)	**0.0968** (± 0.0002)	4.0%
	Recall@10	0.0402	0.1077	0.0950	0.1059	0.0826	<u>0.1355</u> (± 0.0002)	0.1125 (± 0.0023)	**0.1410** (± 0.0004)	4.1%
	mAP	0.0229	0.0767	0.0711	0.0764	0.0580	<u>0.1022</u> (± 0.0003)	0.0811 (± 0.0019)	**0.1073** (± 0.0002)	5.0%

Table 2: Evaluation results with our model and other models. The top-ranked and the second-ranked results are highlighted in bold and underlined, respectively. We use the results of POP, BPR, FMC, FPMC, GRU4Rec from the Caser paper [34]. We additionally reported confidence intervals for top-three models, i.e. Caser, SASRec, and SelCa.

As shown in Table 3, the performance of our model without self-attention significantly decreased in the MovieLens dataset compared to the Gowalla dataset. It is because the MovieLens dataset has strong sequential trends than the Gowalla dataset.

The categorical preference gate improves the model performance in both datasets. This shows that the gate combines two vectors efficiently and also prevents overfitting by reducing the number of parameters in the last fully connected layer. In addition, the category embedding module plays an important role in Gowalla dataset, while self-attention plays an important role in MovieLens dataset. We infer that, because Gowalla is a location-based social check-in dataset, behaviors of users are influenced by global social trends rather than by sequential trends. These findings also are shown in Fig. 4. In MovieLens dataset, as the length of input sequences increases, the performance increases but, it decreases in Gowalla dataset.

We use the LDA to obtain the category probabilities for the given item sequence. Although matrix factorization (MF)-based method such as NMF [20] also can be used in our model, the results using NMF instead of LDA show that performances using the MF are lower than with the LDA. This is because the MF is not a probabilistic-based method and therefore the sum of topic weights from a matrix factorization based method may exceed 1, which makes the combined category embedding vector out of the embedding space of our trained model.

Interestingly, we observe a decrease in model performance when the user noise vector is excluded. From this, we conclude that the noise vector also prevents overfitting. Furthermore, the random noise vector is linearly transformed before being added to the user embedding vector in our model as we explained in Section 3.3. Directly adding the random noise vector to the user embedding vector may hamper the performance by distorting the user information. Thus we propose a technique to first apply a linear transformation to the noise vector. This prevents not only the user information loss but also overfitting. In the absence of the linear transformation, performances are significantly deteriorated as shown in Table 3, e.g., 17.6% and 86.3% lower for MovieLens and Gowalla, respectively.

5.3 Visualization of Item Embedding

In order to investigate whether each item embedding vector is trained appropriately or not we visualize a scatter plot using item embedding vectors. In order to visualize each d-dimensional embedding vector into 2-dimensional space, we apply dimension reduction using the uniform manifold approximation and projection model

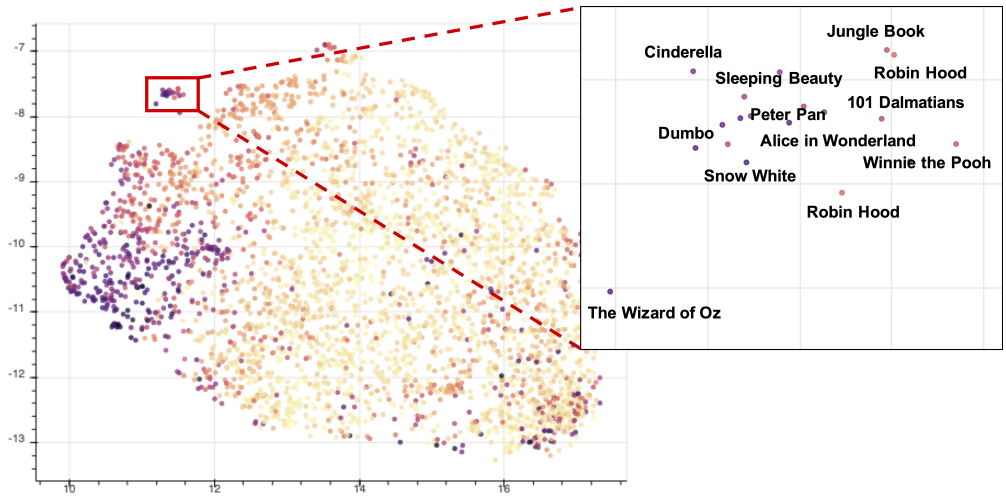


Figure 5: Scatter plot of item embedding vectors using UMAP. We have darkened the circle of the movie as it is older.

Model	MovieLens	Gowalla
SelCa	**0.1967**	**0.1073**
SelCa w/o SA	0.0580	0.1028
SelCa w/o CE	0.1921	0.0831
SelCa w/o CPG	0.1899	0.0836
SelCa w/o UN	0.1889	0.1068
SelCa w/o NL	0.1673	0.0576
SelCa With NMF	0.1934	0.1058

Table 3: Ablation test using mAP score. Each row represents the full model, the model without the self-attention module (SA), the model without the category embedding module (CE), the model substituted concatenation for the categorical preference gate (CPG), and the model without the user noise vector (UN), the model without the linear transform on user noise vector (NL), the model substituted NMF for LDA, respectively.

(UMAP) [24] which more effectively preserves the local and global structure in a low-dimensional embedding space with faster time complexity than t-SNE [22]. After conducting dimension reduction for all item embedding vectors, we draw scatter plot and the result is shown in Fig. 5. We have darkened the circle of the movie as it is older.

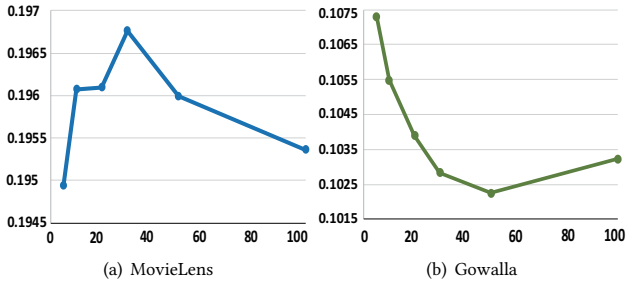
Interestingly, old movies generally tend to be located outside while recent movies tend to be located in the middle. This is natural that films released in the same era reflect the social aspects of the times, so they tend to be similar. In addition, as shown in the zoomed region in Fig. 5, movies with similar genres such as *Cinderella*, *Jungle Book*, and *Snow White*, are located close to each other. From these findings, we conclude that our model effectively captures additional information although we do not provide meta information such as movie release date and genre.

5.4 Analysis of Category Embedding

We evaluate mAP scores with the number of categories for each dataset in order to find the optimal number of latent categories for each dataset. We conduct experiments with 5, 10, 20, 30, 50, 100 categories. The optimal numbers of categories in MovieLens and Gowalla are 30 and 5, respectively. As we find the characteristic difference between the two datasets, the trends for the two datasets

Title	Genre
March of the Wooden Soldiers	Commedy
How I Won the War	Commedy, War
The Return of the Pink Panther	Commedy
Speed	Action, Romance, Thriller
Tomorrow Never Dies	Action, Romance, Thriller
The Crying Game	Drama, Romance, War
The Man Who Knew Too Much	Thriller
Secret Agent	Thriller
No Way Out	Thriller
Face/Off	Action, Sci-Fi, Thriller
Independence Day	Action, Sci-Fi, War
Superman	Action, Adventure, Sci-Fi

Table 4: Examples extracted using several category embedding vectors. For each row, we report the top-three predicted movies using the same category embedding vector.



(a) MovieLens (b) Gowalla

Figure 6: mAP scores with the number of categories for each dataset. The average confidence intervals for MovieLens and Gowalla are ±0.0005, ±0.0007, respectively.

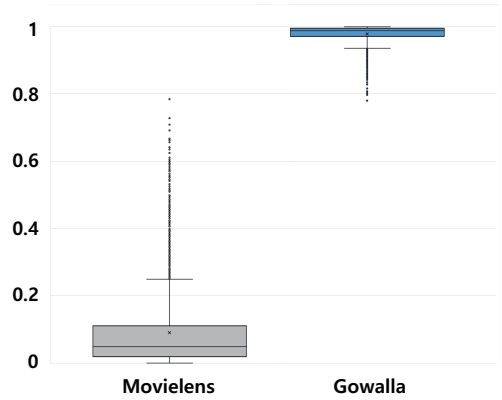


Figure 7: Visualization using the statistics of the categorical preference gate scores with box plots for each dataset.

are also different. As the number of categories increases until 30, performance in MovieLens increases but decreases after. On the other hand, as the number of categories increases, performance in Gowalla dataset decreases until 50 but increases after. We infer that there exists a trade-off between the effect of category embedding and overfitting. In future work, we are going to apply further elaborate topic modeling models, such as the hierarchical Dirichlet process (HDP) [35], that have the capability of automatically finding the optimal number of topics.

We further conduct qualitative analysis to investigate the effects of category embedding. We first replace the content vector $\mathbf{v}_{content}$ with each category embedding vector $\mathbf{e}_c^{cat}$ and set the user embedding vector $\mathbf{v}_u$ as the zero vector, and then we predict the target items. The results are shown in Table 4 using four categories. In each row, we listed the top three movies predicted using the same category embedding vector $\mathbf{e}_c^{cat}$. In the table, movies with similar genre tend to be clustered in each category. Note that we do not give any genre information to our model. This suggests that our category embedding provides item-collective information suitable for the user's preference.

5.5 Analysis of Categorical Preference Gate

Previously, we introduced the categorical preference gate that effectively combines the categorical vector and the self-attentive vector. To investigate the general characteristics of the categorical preference gate, we conduct several experiments and analyze the categorical preference gate.

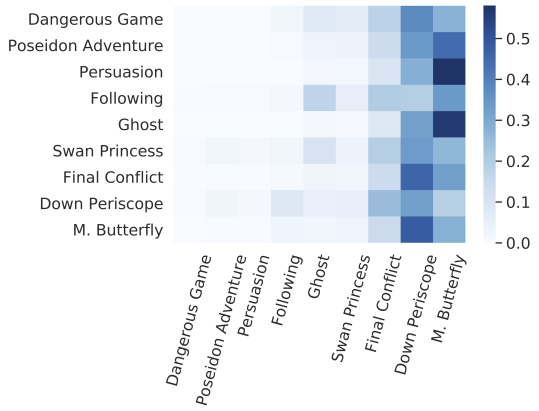
First of all, we select two users that have the highest and the lowest gate scores from the categorical preference gate with their recent movie sequence. The recent movies and their genres corresponding to users with the highest and the lowest gate scores are shown in Table 5. From this, we deduce the tendencies of the categorical preference gate. In the examples with the lowest gate scores, our model utilizes sequential information, e.g., recently viewed movies, more than the global information. On the other hand, when the user watches movies with different genres, the gate value tends to be high. In this case, our model assigns a large weight to the categorical embedding vector, which means our model exploit global information, e.g., a user's general preference or a mixture of genre information combined from recently viewed movies, much more

User 1 (0.7839)	**User 2** (0.0001)
Forrest Gump (Comedy, Romance, War)	Alien (Horror, Sci-Fi, Thriller)
RoboCop (Action, Crime, Sci-Fi)	Psycho (Horror, Thriller)
Schindler's List (Drama, War)	The Exorcist (Horror)
Honey, I Shrunk the Kids (Adventure, Comedy, Fantasy)	The Shining (Horror)
The Craft (Drama, Horror)	Invasion of the Body Snatchers (Horror, Sci-Fi)
Die Hard: With a Vengeance (Action, Thriller)	Body Heat (Crime, Thriller)
Runaway Bride (Comedy, Romance)	Vertigo (Mystery, Thriller)
Star Wars IV (Action, Adventure, Sci-Fi)	Seven Days in May (Thriller)
Now and Then (Drama)	Die Hard (Action, Thriller)
Braveheart (Action, Drama, War)	Chinatown (Mystery, Thriller)

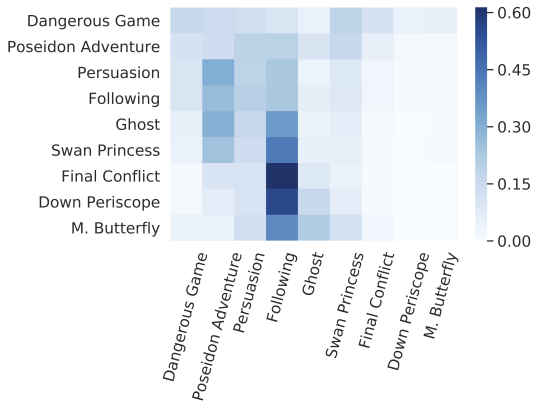
Table 5: Movies and their genres from the users that have the highest and lowest categorical preference gate scores.

than sequential information. This suggests that the use of global information is reasonable because if the history sequence from the user is inconsistent, the sequential history lacks information. Thus, by applying the categorical preference gate to our model, our model has the ability to reflect global trends and information such as the user's general preference and genre characteristics.

In Table 4, we show the experimental results of mAP scores along the sequence length. In MovieLens dataset, mAP scores increase as the sequence length increases, whereas mAP scores decrease as the sequence length increases in Gowalla dataset. This means that sequential information is much more crucial in MovieLens dataset than in Gowalla dataset. To find the differences between the two datasets, we calculate the statistics of the categorical preference gate score for each dataset and draw box plot figures using the gate scores. The box plots of the gate scores in Fig. 7 show a different tendency for each dataset. In the MovieLens dataset, the gate values are generally distributed closely to zero. This means the self-attentive vectors play a major role in MovieLens. On the contrary, as the gate scores of Gowalla are mostly close to one. From this, it is evident that the categorical vector plays an important role in the Gowalla dataset. We believe that this tendency is due to the characteristics of each dataset. For MovieLens, people tend to decide movies they will watch, based on the recent movies they have seen. Therefore, the MovieLens dataset has a high degree of relevance in sequential order. On the other hand, the Gowalla dataset, which is a location-based social dataset, has a low degree of relevance in sequential order.



(a) Attention weights from the first attention head



(b) Attention weights from the second attention head

Figure 8: Heat maps with attention weights for each attention head in the last self-attention layer.

From this, we conclude that our model provides high performance consistently by exploiting sequential and global information and combining these two information effectively, although two datasets have different tendencies and characteristics.

5.6 Visualization of the Self-Attention Score

In order to analyze the characteristics of self-attention module, we visualize head maps with attention weights. After packing item embedding vectors $\{\mathbf{e}_1^{item}, \cdots, \mathbf{e}_{|\mathcal{L}|}^{item}\}$ into $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, our self-attention module calculates attention weights $\mathbf{A}_{i,j}$ corresponding to the key $\mathbf{k}_j \in \{\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_{|\mathcal{L}|}\}$ as

$$\mathbf{A}_{i,j} = \frac{exp\left(\mathbf{q}_i^\top \mathbf{k}_j\right)}{\sum_n exp\left(\mathbf{q}_i^\top \mathbf{k}_n\right)}. \tag{15}$$

Note that the attention scores with the item itself can be low because each query and key vector is linearly transformed through different fully-connected layers before calculating the attention scores. In addition, because we apply multi-head attention with n_{head} heads and n_{att} attention layers, our self-attention module calculates $n_{att} \times n_{head}$ attention weight matrices. For MovieLens, the optimal number of attention heads is two as we set in Section 4, and therefore we extract two attention weight matrices from the last attention layer and then draw heatmap figures using these attention weight matrices.

In Fig. 8, the attention score is high when the query and the key movies are similar to each other. For instance, the attention scores for query movies *Persuasion* and *Ghost* with the key movie *M. Butterfly* are high. In addition, the first attention head mainly focuses on the last three items, whereas the second attention head mainly focuses on the first items. This demonstrates that multi-head attention in our self-attention module effectively utilizes the sequential information by separating the former part and the latter part from the input item sequence and analyzing the two parts jointly.

Trial	User 1	User 2
1	True Lies Con Air Mission: Impossible Twister Conan the Barbarian The Ghost and the Darkness Armageddon Stargate Clear and Present Danger Willow	Driving Miss Daisy Parenthood Good Morning, Vietnam The Rainmaker A League of Their Own The Karate Kid Dances with Wolves A Few Good Men The Color Purple The Outsiders
2	True Lies Mission: Impossible Con Air Twister Conan the Barbarian Armageddon The Ghost and the Darkness Clear and Present Danger Willow The Rock	Driving Miss Daisy Parenthood Good Morning, Vietnam The Rainmaker A League of Their Own The Karate Kid Dances with Wolves A Few Good Men The Outsiders Fried Green Tomatoes
3	True Lies Con Air Mission: Impossible Twister Conan the Barbarian Armageddon Star Wars: Episode I The Ghost and the Darkness Clear and Present Danger The Rock	Driving Miss Daisy Parenthood Good Morning, Vietnam The Rainmaker The Karate Kid A League of Their Own Dances with Wolves A Few Good Men The Outsiders The Color Purple

Table 6: Results of diverse recommendations. We randomly select two users and extract the top-10 predicted items from the corresponding users three times.

5.7 Diverse Recommendation

Our model has the ability to provide diverse recommendations by adding the linearly transformed noise vector z to the user embedding vector e_u^{user}. In order to investigate the effects of the diverse

recommendation, we randomly select two users and predict the next items three times. As the results with the top-10 predicted movies for each user are reported in Table 6, highly ranked items remain unchanged even with the noise vector, while relatively lower-ranked items tend to change. Moreover, movies such as *Star Wars* and *Fried Green Tomatoes* are newly recommended. This shows our model provides diverse recommendations while preventing deterioration of performance. Therefore, we conclude that our model has the ability to recommend diverse items to users while preventing loss in user information such as user's general preference and overfitting problem as shown in Fig. 3.

6 CONCLUSIONS

In this paper, we proposed a self-attentive sequential recommender system using the global structure as a novel approach in order to exploit global information in the process of sequential recommendation. Our model is capable of effectively exploiting both short-term sequential patterns and general preference from the entire user-item interaction data. With this architecture and a novel approach, our model outperforms other state-of-the-art models. Furthermore, by applying diverse recommendation using a random noise vector followed by a linear transform, our model supports users in exploring diverse choices while mitigating user information loss and overfitting. Our proposed approach can also be applied to various tasks, such as recommender system, dialogue system, and personal assistant, that deal with both sequential and global information.

7 ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT (2017M3C4A7063570).

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *ACM SIGMOD*. 207–216.
[2] R. Agrawal and R. Srikant. 1995. Mining sequential patterns. In *ICDE*. 3–14.
[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *JMLR* (2003), 993–1022.
[4] Chenwei Cai, Ruining He, and Julian McAuley. 2017. SPMC: socially-aware personalized markov chains for sparse sequential recommendation. In *IJCAI*. 1476–1482.
[5] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2017. Natural language inference with external knowledge. *arXiv:1711.04289* (2017).
[6] Wen-Yen Chen, Dong Zhang, and Edward Y Chang. 2008. Combinational collaborative filtering for personalized community recommendation. In *KDD*. 115–123.
[7] Disheng Dong, Xiaolin Zheng, Ruixun Zhang, and Yan Wang. 2018. Recurrent Collaborative Filtering for Unifying General and Sequential Recommender.. In *IJCAI*. 3350–3356.
[8] Songjie Gong. 2010. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *Journal of Software* (2010), 745–752.
[9] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*. 161–169.
[10] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*. 191–200.
[11] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. In *IJCAI*. 2227–2233.
[12] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.
[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv:1511.06939* (2015).
[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.
[15] Tero Karras, Samuli Laine, and Timo Aila. 2018. A style-based generator architecture for generative adversarial networks. *arXiv:1812.04948* (2018).
[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
[17] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv:1312.6114* (2013).
[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer Society* (2009), 30–37.
[19] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. 1188–1196.
[20] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*. 556–562.
[21] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.
[22] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* (2008), 2579–2605.
[23] Benjamin M Marlin. 2004. Modeling user rating profiles for collaborative filtering. In *NIPS*. 627–634.
[24] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426* (2018).
[25] Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. 2018. An attentive interaction network for context-aware recommendations. In *CIKM*. 157–166.
[26] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *ACL*. 130–136.
[27] Shuzi Niu and Rongzhi Zhang. 2017. Collaborative sequence prediction for sequential recommender. In *CIKM*. 2239–2242.
[28] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *ICDM*. 502–511.
[29] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David MJ Tax. 2017. Interacting attention-gated recurrent networks for recommendation. In *CIKM*. 1459–1468.
[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[31] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.
[32] Hanhuai Shan and Arindam Banerjee. 2010. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*. 1025–1030.
[33] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In *CIKM*. 127–136.
[34] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *ICWSM*. 565–573.
[35] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical Dirichlet processes. In *NIPS*. 1385–1392.
[36] Lyle H Ungar and Dean P Foster. 1998. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*. 114–129.
[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[38] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. 1235–1244.
[39] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *WWW*. 3307–3313.
[40] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *ICDM*. 495–503.
[41] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *ICWSM*. 153–162.
[42] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention networks. In *IJCAI*. 3926–3932.
[43] Chenyi Zhang, Ke Wang, Hongkun Yu, Jianling Sun, and Ee-Peng Lim. 2014. Latent factor transition for dynamic collaborative filtering. In *SDM*. 452–460.
[44] Qi Zhang, Jiawen Wang, Haoran Huang, Xuanjing Huang, and Yeyun Gong. 2017. Hashtag Recommendation for Multimodal Microblog Using Co-Attention Network.. In *IJCAI*. 3420–3426.
[45] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next Item Recommendation with Self-Attention. *arXiv:1808.06414* (2018).
[46] Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-aware tensor-based recommendation. In *CIKM*. 1153–1162.