Pattern-enhanced Contrastive Policy Learning Network for Sequential Recommendation

Xiaohai Tong[1] , **Pengfei Wang**[2*] , **Chenliang Li**[3*] , **Long Xia**[4] and **Shaozhang Niu**[1]
[1]Beijing Key Laboratory of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing, China
[2]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
[3]Wuhan University, Wuhan, China
[4]Baidu Inc., Beijing, China
wangpengfei@bupt.edu.cn, cllee@whu.edu.cn

Abstract

Sequential recommendation aims to predict users' future behaviors given their historical interactions. However, due to the randomness and diversity of a user's behaviors, not all historical items are informative to tell his/her next choice. It is obvious that identifying relevant items and extracting meaningful sequential patterns are necessary for a better recommendation. Unfortunately, few works have focused on this sequence denoising process.

In this paper, we propose a Patte***R***n-enhanced Contr***A***stive ***P***olicy Learning Network (RAP for short) for sequential recommendation, RAP formalizes the denoising problem in the form of Markov Decision Process (MDP), and samples action for each item to determine whether it is relevant with the target item. To tackle the lack of relevance supervision, RAP fuses a series of mined sequential patterns into the policy learning process, which work as a prior knowledge to guide the denoising process. After that, RAP splits the initial item sequence into two disjoint subsequences: a positive subsequence and a negative subsequence. At this, a novel contrastive learning mechanism is introduced to guide the sequence denoising and achieve preference estimation from the positive subsequence simultaneously. Extensive experiments on four public real-world datasets demonstrate the effectiveness of our approach for sequential recommendation.

1 Introduction

Sequential recommendation, which aims to recommend the next few items that the user will likely interact with in the near future, now is playing an increasingly important role in various application scenarios [Zhang *et al.*, 2019b]. For this task, sequential dependency has been proven to be an important factor since items interacted in future are largely dependent on the previous ones [Kang and McAuley, 2018]. However, due to the randomness of the user's behaviors and his/her diverse interests, not all historical items contain useful signals w.r.t. the preference estimation for the target item. When taking these irrelevant yet noisy items into consideration, it becomes hard to precisely distill preference information for better recommendation performance. A good recommender should be able to extract these credible sequential dependencies relevant with the target item. Here, in this paper, we refer to this process as a sequence denoising task. Although it is appealing in theory, limitation and complication of user-item interactions make the denoising process non-trivial.

Over the past decades, there have been many research efforts related to sequence denoising in one form or another. Earlier works, including association rule [Lin *et al.*, 2002] and sequential pattern mining [Mooney and Roddick, 2013], can be casted as a particular denoising process by extracting plausible patterns. With the prosperity of deep neural network, RNN-based and CNN-based models have become two mainstreams to distill discriminative information for recommendation. These methods mostly have a deficiency of treating each item equally, which may introduce much noise into the model learning process. Later, attention-based approaches are proposed [Kang and McAuley, 2018; Yu *et al.*, 2019] to estimate an importance weight for each item for better recommendation. Though effective, the existence of noisy data may still complicate the preference learning process. In addition, the interpretability is also limited as they fail to explicitly extract the truly relevant patterns. How to automatically extract credible sequential dependencies relevant with the target item for a correct recommendation, is a crucial and challenging problem. 去噪和权重 作用相同嘛

In this paper, we propose a Patte***R***n-enhanced Contr***A***stive ***P***olicy Learning Network (named RAP) for sequence denoising. Specifically, RAP consists of two modules: a pattern-enhanced policy module and a contrastive learning module. The policy module employs a stochastic policy to sample action for each item from the initial sequence to identify its relevance with the target item. It is worthwhile to note that we have no access to relevance information for a given historical sequence and target item pair. To better facilitate denoising, we further mine a series of sequential patterns, and feed this global prior knowledge into policy network for guidance. Ac-

*Corresponding author.

cording to the sampled actions, RAP produces a positive subsequence and a negative subsequence respectively. It is expected that the positive subsequence containing only the relevant items in the chronological order would ease the preference learning. To achieve effective denoising and preference estimation, a contrastive learning module is then applied on the resultant two subsequences with delayed rewards to make a robust gradient estimation. Empirical results on four real-world datasets demonstrate that the process of sequence denoising is feasible and our proposed method can outperform many state-of-the-art baselines significantly. We provide detailed analysis on the proposed model, and conduct case studies to gain better understanding on the learned patterns.

2 Related Work

In this section, we briefly review three lines of related work, *i.e.,* sequential recommendation, RL-based recommendation and contrastive learning.

2.1 Sequential Recommendation

Sequential recommendation aims to predict users' future behaviors given their historical interaction data. Some earlier sequential recommendation methods rely on item-item transition matrices to capture the sequential patterns in the user interaction sequence [Rendle *et al.*, 2010; Cheng *et al.*, 2013]. Due to the nonlinear expressive capacity, RNN-based and CNN-based models have been widely utilized to model the interaction sequences. Specifically, RNN-based models focus on exploiting sequential dependencies from interaction sequences for recommendation [Hidasi *et al.*, 2016; Huang *et al.*, 2018], while the CNN-based methods allow feature composition over the consecutive behaviors in interaction sequence [Tang and Wang, 2018].

Though effective, none of the above works have introspected the rationality of fusing all historical interactions into account for the preference estimation. Recently, the attention mechanism has shown promising potential to support a context-aware preference learning for recommendation [Wang *et al.*, 2018; Kang and McAuley, 2018]. However, these approaches still fail to obtain the truly relevant items for further consideration, which limits the recommendation performance and interpretability.

2.2 RL-based Recommendation

Reinforcement Learning aims to find an optimal action in a particular situation that would eventually maximize the long-term outcome. It has been introduced into recommender systems and achieved great success as its advantage of considering users' long-term feedbacks [Zhao *et al.*, 2018b; Zou *et al.*, 2020]. For example, [Zhao *et al.*, 2018a] propose a deep reinforcement learning model to automatically learn the optimal strategies for page-wise recommendation. [Xian *et al.*, 2019] propose a policy-gradient approach to extract paths from Knowledge Graph(KG), and take these paths as the recommendation interpretation. [Zheng *et al.*, 2018] design a DQN-based reinforcement learning framework to make online personalized news recommendation. [Zou *et al.*, 2019] formulate the ranking process as a multi-agent Markov Decision Process, where mutual interactions between documents are incorporated. [Zhang *et al.*, 2019a] propose a hierarchical reinforcement learning to revise user profile for course recommendation. [Xin *et al.*, 2020] propose a self-supervised reinforcement learning framework for sequential recommendation. Recently, the contextual information is also considered to enhance the reinforcement learning process for recommendation. For instance, [Zhou *et al.*, 2020] investigate the potential of leveraging KG to improve the sample efficiency of RL methods.

2.3 Contrastive Learning

Contrastive learning has recently achieved great success in various domains [Dai and Lin, 2017; Laskin *et al.*, 2020]. The key idea of contrastive learning is to perform unsupervised learning over semantically similar (positive) and dissimilar (negative) instance pairs. By encouraging the representation of similar pairs to be close, and those of dissimilar pairs to be far apart [Chuang *et al.*, 2020], contrastive learning can alleviate the extensive demand for human annotations, and enable mining the underlying correlations behind the rich structure inside data [Devlin *et al.*, 2019]. Inspired by these advances, we present the first contrastive learning based sequence denoising model for sequential recommendation.

3 Problem Definition

Let $\mathcal{U}$=$\{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$ denote a set of users, $\mathcal{E}$=$\{e_1, e_2, \cdots, e_{|\mathcal{E}|}\}$ be a set of items, where $|\mathcal{U}|$ and $|\mathcal{E}|$ represent the total number of unique users and items, respectively. For each user $u \in \mathcal{U}$, we use e^u=$\{e_1^u, e_2^u, \cdots, e_n^u\}$ to denote his/her interaction sequence, where $e_i^u \in \mathcal{E}$ is the item that u has interacted with at time step i and n is the length of the interaction sequence e^u.

In this paper, we consider an episodic RL approach to address the sequence denoising for sequential recommendation. Given a target item e_t, at each time step i, the process is in some state $s_i \in S$. According to state s_i, the agent performs an action a_i modeled by a policy $\pi(a_i|s_i)$. The action space is $a_i \in \{positive = 1, negative = 0\}$, where we use positive action to indicate that the item is relevant with the target item, and a negative action indicates that the item should be removed from the initial sequence. Given the definition of states and actions, the probability of choosing a_i can be written as follows:

$$\pi(a_i = 1|s_i) = \sigma(\mathbf{s}_i \mathbf{w}_1 + b) \tag{1}$$

where $\mathbf{s}_i$ is the representation of state s_i, $\mathbf{w}_1$ and b denote parameters of the policy module. After sequence denoising, we can obtain a positive subsequence e^+ and a negative subsequence e^- from the initial sequence e^u. The subsequence e^+ is a chronologically ranked list of all relevant items. In contrast, e^- is the list of all irrelevant items. Our aim is to infer the likelihood that the user will interact with e_t based on subsequence e^+.

4 Pattern-enhanced Contrastive Policy Learning Network

In this section, we introduce our Pattern-enhanced Contrastive Policy Learning Network (RAP) in detail. Figure 1

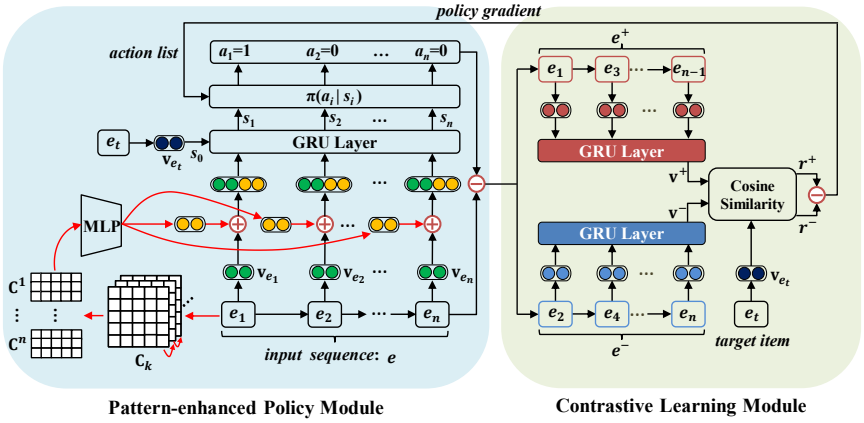

Figure 1: The overall architecture of the proposed RAP.

shows the architecture of RAP, which consists of two main modules: 1) a pattern-enhanced policy module to sample action for each item in the initial interaction sequence, and split the initial sequence into two subsequences, which are expected to keep relevant or irrelevant items respectively; 2) a contrastive learning module is further applied on these two subsequences to generate a contrastive loss to guide the model optimization. For simplicity, we describe algorithmic details for a single user u, and it is straightforward to extend the formulas to a set of users. For brevity, we will omit the superscript u in the notations.

4.1 Pattern-enhanced Policy Module

RAP utilizes a policy module to determine historical items' relevance with the target item. However, running a complicated policy on a sequence with sparse user-item interactions is not easy. In addition, this process is not guaranteed since no item-level relevance information is provided to supervise the denoising process. Considering the collective wisdom (*i.e.,* sequential patterns) is well expressed in massive user behaviors, we choose to mine sequential patterns, and treat them as prior knowledge to guide the denoising process.

Specifically, we first use SPADE [Zaki, 2001] to extract the eligible patterns from the interaction sequences. We compile the extracted k-length patterns into a matrix $\mathbf{C}_k$, where $\mathbf{c}_k^i$ represents the i-th row of $\mathbf{C}_k$, and the j-th element of $\mathbf{c}_k^i$ is the number of patterns that start from e_i and end with e_j. Given an item e_i, we use $\mathbf{C}^i = [\mathbf{c}_1^i, \mathbf{c}_2^i, ..., \mathbf{c}_k^i]$ to express the corresponding pattern information, and further feed it into the calculation of policy state s_i to enrich its semantic:

$$\boldsymbol{s}_i = \mathbf{GRU}\big[\boldsymbol{s}_{i-1}, I(a_i = 1) \cdot (\mathbf{v}_{e_i} \oplus \mathbf{MLP}(\mathbf{C}^i))\big] \quad (2)$$

where $\mathbf{GRU}(\cdot)$ is the Gated Recurrent Unit (GRU), $\mathbf{MLP}(\cdot)$ represents a multi-layer perception, $\mathbf{v}_{e_i}$ is the embedding vector of item e_i, $I(\cdot)$ is an indicator function, and $\oplus$ is a concatenation operator.

After sampling a_i with Equation (1), we can decide whether to keep the item for preference estimation. Specifically, when $a_i = 1$, the item is relevant and retained. On the other hand, $a_i = 0$ suggests the opposite. In the latter case, s_i is set to be the same as s_{i-1}. Note that when $i = 0$, the state is initialized as $s_0 = \mathbf{v}_{e_t}$, where e_t is the target item. In this sense, we inject the target item e_t into policy component to enable a context-aware denoising process.

By feeding the pattern information into the state representation, our policy can foresee future sequential information when making a decision, which could offer insightful clues to determine item-level relevance without direct supervision signals.

4.2 Contrastive Learning Module

Given an interaction sequence e, we use $\pi(a_i|s_i)$ to sample an action for each constituent item sequentially. That is, we can obtain an action list $(a_1, \cdots, a_n)$. In this way, we naturally separate the initial sequence e into two subsequences: positive subsequence e^+ and negative subsequence e^- as aforementioned. Specifically, given a sampled action list $(a_1, \cdots, a_n)$, the probability of generating e^+ can be calculated as follows:

$$P(e^+) = \prod_i \pi(a_i|s_i) P(s_{i+1}|s_i, a_i) = \prod_i \pi(a_i|s_i)$$

Moreover, based on the series of sampled e^+, the objective function can be written as follows:

$$\begin{aligned} L = \mathbb{E}_{\pi \sim (a_1 \cdots a_n)} r^+ &= \sum_{(a_1 \cdots a_n)} P(e^+) r^+ \\ &= \sum_{(a_1 \cdots a_n)} \prod_i \pi(a_i|s_i) r^+ \end{aligned}$$

where r^+ is the delayed reward over e^+. As e^+ is expected to precisely capture informative semantics that are relevant with the target item e_t, a simple solution is to encode positive subsequence e^+. Here, we choose another GRU to derive the subsequence representation $\mathbf{v}^+$ as the hidden state of the last item in e^+. Then, we use cosine similarity between e^+ and e_t as the reward:

$$r^+ = \frac{\mathbf{v}^+ \cdot \mathbf{v}_{e_t}}{|\mathbf{v}^+| \cdot |\mathbf{v}_{e_t}|} \quad (3)$$

However, this strategy seems not an optimal option as the optimization does not utilize irrelevant items in e, which also aggravates the sparse problem. In addition, due to the lack of supervision signals, a wrong distinction of our policy module (*i.e.*, determining a relevant item as irrelevant) will also adversely affect performance. To fully utilize both relevant and irrelevant items, inspired by [Chen *et al.*, 2018], we introduce a contrastive learning process to enhance model learning.

Here, we further adopt another GRU to encode the negative subsequence as $\mathbf{v}^-$. The corresponding probability of generating negative subsequence e^- is as follows:

$$P(e^-) = \prod_i \big(1 - \pi(a_i|s_i)\big)\big(1 - P(s_{i+1}|s_i, a_i)\big)$$
$$= \prod_i (1 - \pi(a_i|s_i))$$

Similarly, r^- is calculated as the cosine similarity between $\mathbf{v}^-$ and $\mathbf{v}_{e_t}$. The final objective function is rewritten as follows:

$$L = \sum_{(a_1 \cdots a_n)} \prod_i \pi(a_i|s_i) r^+ - \sum_{(a_1 \cdots a_n)} \prod_i \big(1 - \pi(a_i|s_i)\big) r^-$$

4.3 Learning and Prediction

We use the Adam optimizer to maximize the objective function over all training instances. With the learned RAP model, given the interaction sequence of a user and the candidate items, we first scan the whole sequence according to Equation (2), and select each action with the maximal probability, which can be written as follows:

$$a_i^* = \arg\max_{a_i} \pi(a_i|\boldsymbol{s}_i) \tag{4}$$

By this we extract the truly relevant items, and calculate its reward r^+ according to Equation (3). We then rank all candidate items according to their rewards, and return the top-N as the final recommendations.

5 Experiments

In this section, we evaluate our proposed RAP[1]against several state-of-the-art methods. We first describe the datasets, baseline methods, evaluation metrics used in the experiments, and then analyze the experimental results.

5.1 Experimental Setup

Datasets

We conduct experiments on four publicly available real-world datasets from three different domains. *Beauty* and *Games* are two different categories of Amazon dataset[2], which span from May 1996 to July 2014. *LastFM*[3] is a music listening dataset released from Last.fm online music system and we take a subset which contains interactions from Jan 2015 to June 2015. *MovieLens* is a popular benchmark dataset collected from the MovieLens website. In this work, we adopt ML-1m[4] which has one million user-movie interactions.

[1]The implementation of our model is available at https://github.com/heilsvastika/RAP

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

[3]http://www.cp.jku.at/datasets/LFM-1b/

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

Datasets	#users	#items	#actions	Avg. length
Beauty	52,204	57,289	0.4M	7.6
Games	31,013	23,715	0.3M	9.3
LastFM	7,694	30,658	0.2M	21.11
ML-1m	6,040	3,416	1.0M	163.5

Table 1: The statistics of the datasets.

Preprocessing

We follow the same preprossessing procedure in [Rendle *et al.*, 2010; He and McAuley, 2016]. For all datasets, we use timestamps to order the interactions of each user. We discard users and items with fewer than 5 interactions. For partitioning, we split the historical sequence for each user *u* into three parts: (1) the most recent interaction for testing; (2) the next-to-last interactions for validation; and (3) all remaining interactions for training. This *leave-one-out* evaluation strategy has been widely used in related works [Kang and McAuley, 2018; Tang and Wang, 2018]. The detailed statistics for the four datasets are reported in Table 1.

Baseline Methods

We adopt the following representative and state-of-the-art methods as baselines for performance comparison, including non-sequential models and sequential models:

For non-sequential models, we consider BPR and NCF:

- **BPR-MF**: The Bayesian Personalized Ranking based Matrix Factorization [Rendle *et al.*, 2009] is a conventional method for learning pairwise personalized rankings from user implicit feedback.
- **NCF**: The Neural Collaborative Filtering [He *et al.*, 2017b] models user-item interactions with a series of nonlinear transformations.

For sequential models, we consider both shallow and deep models:

- **FPMC**: It captures users' general taste as well as their sequential behaviors by combining MF with first-order Markov Chains (MC) [Rendle *et al.*, 2010].
- **GRU4Rec**: It uses GRU with ranking based loss to model user sequences for session-based recommendation [Hidasi *et al.*, 2016].
- **Caser**: It employs CNN in both horizontal and vertical way to model high-order Markov Chains for sequential recommendation [Tang and Wang, 2018].
- **NARM**: It employs RNNs with attention mechanism to capture the users' preference and sequential behaviors [Li *et al.*, 2017].
- **SASRec**: It uses a left-to-right Transformer model to capture users' sequential behaviors for recommendation [Kang and McAuley, 2018].
- **HRL**: It utilizes a hierarchical reinforcement learning algorithm [Zhang *et al.*, 2019a] to revise the user profiles and tune the recommendation model on the revised profiles.

Datasets	Metric	(a) BPR	(b) NCF	(c) FPMC	(d) GRU4Rec	(e) Caser	(f) NARM	(g) SASRec	(h) HRL	(i) SASRec$^+$	(j) BERT4Rec	(k) RAP	Improv.
Beauty	HR@10	18.80	20.17	23.48	25.85	30.54	31.71	33.50	35.28	35.66	<u>36.01</u>	**37.43**	3.94
	NDCG	12.11	13.05	15.65	17.10	18.25	19.39	21.37	21.60	22.43	<u>22.70</u>	**23.75**	4.63
Games	HR@10	25.20	29.46	33.50	36.72	45.21	48.61	56.27	56.165	57.12	<u>58.39</u>	**60.41**	3.46
	NDCG	19.74	21.98	22.67	25.45	34.09	37.20	40.84	41.31	42.15	<u>42.69</u>	**44.61**	4.50
LastFM	HR@10	19.41	23.28	25.78	28.29	31.23	32.53	34.65	34.97	35.44	<u>36.47</u>	**37.63**	3.18
	NDCG	13.83	15.37	15.98	19.05	18.99	20.63	23.72	21.59	24.73	<u>25.12</u>	**26.28**	4.62
ML-1m	HR@10	28.61	26.60	36.13	42.54	51.89	59.29	62.01	62.30	62.59	<u>63.04</u>	**64.35**	2.08
	NDCG	22.63	19.36	29.37	34.51	42.35	47.52	51.62	49.35	52.43	<u>52.76</u>	**55.18**	4.59

Table 2: Performance comparison on sequential recommendation between the baselines and our model (all the values in the table are percentage numbers with % omitted). The best performance in each row is in bold font, and the underlined scores represent best baseline performance. The last column shows the relative improvement of our results against the best baseline, which is significant at p-value$\leq$ 0.05.

- **SASRec$^+$**: We design an extension of SASRec by directly deleting the items with their attention scores smaller than a given threshold. Comparing with SASRec, in SASRec$^+$, items with smaller attention scores have no influences towards preference learning.
- **BERT4Rec**: It employs the deep bidirectional self-attention to model user sequences and achieves state-of-the-art recommendation performance [Sun *et al.*, 2019].

For NCF, GRU4Rec, Caser, SASRec and BERT4Rec, we use the implementations released by the corresponding authors. FPMC and NARM are implemented by RecBole [Zhao *et al.*, 2020]. For the rest, we implement them using PyTorch.

Evaluation Metrics

To conduct the performance evaluation, we adopt two widely used metrics: Hit Ratio(HR) and NDCG [He *et al.*, 2017b; He *et al.*, 2017a]. Considering we only have one test item for each user, HR@N is equivalent to Recall@N and proportional to Precision@N. In this work, we report HR and NDCG with $N = 10$. With paired t-test, performance differences are considered statistically significant with 0.05 level. We pair each ground-truth item in the test set with $1,000$ randomly negative items that the user has not interacted with, and rank these items with the ground-truth item together [Krichene and Rendle, 2020].

Implementation Details

We optimize all models with *Adam optimizer* [Kingma and Ba, 2015]. For baselines, we optimize each of them according to the validation sets. For SASRec$^+$, the attention threshold is set to 0.015 also according to the validation sets. As to RAP, the hidden layer size of GRU and the embedding size are set to 100, the learning rate is set to 0.001, and the batch size is 128. The maximum length of sequential patterns extracted by SPADE is 5 (*i.e.,* $k = 5$) when setting the support count to 2.

5.2 Performance Comparison

The overall performance in terms of HR@10 and NDCG@10 is shown in Table 2. Here, we make the following observations:

For the conventional models, we see that the performance of BRP and NCF are relatively poor. It verifies that considering user-item interactions independently is not effective for sequential recommendation. By considering the successive sequential dependencies, FPMC performs better than BPR and NCF.

For the neural models, we see that by adopting deep learning to capture sequential dependencies, these models obtain a better performance than conventional methods. Comparing with GRU4Rec, Caser further considers the union and skip behaviors of sequential patterns, and performs better than the former.

By utilizing an attention mechanism to highlight the most relevant items, we also find that the attention-based sequential models perform better than the none attention-based ones. This observation demonstrates the necessity of discerning the relevance between items for a better recommendation. An interesting observation is that HRL performs better than SASRec, it demonstrates dropping irrelevant items directly is a more effective strategy than assigning smaller weights on these items. Similarly, when removing irrelevant items according to a threshold, SASRec$^+$ performs better than HRL and SASRec. For BERT4Rec, it employs deep bidirectional self-attention and beats all the other baselines on four datasets. Comparing with the attention-based sequential models and the other baselines, our proposed RAP achieves significantly better recommendation performance across the four datasets and two metrics. We believe that the sequence denoising devised in RAP is effective to enhance recommendation performance.

Datasets	Metric	RAP^{-p}_{-c}	RAP^{-p}_{+c}	RAP^{+p}_{-c}	RAP
Beauty	HR@10	34.56	35.69	35.38	**37.43**
	NDCG@10	19.56	20.23	20.01	**23.75**
Games	HR@10	57.08	58.23	58.04	**60.41**
	NDCG@10	39.03	42.46	42.04	**44.61**
LastFM	HR@10	34.79	36.05	35.66	**37.63**
	NDCG@10	21.31	25.57	25.35	**26.28**
ML-1M	HR@10	62.12	63.13	63.30	**64.35**
	NDCG@10	49.03	53.09	53.32	**55.18**

Table 3: Performance comparison of RAP and its three sub-models over four datasets. (all the values in the table are percentage numbers with% omitted)

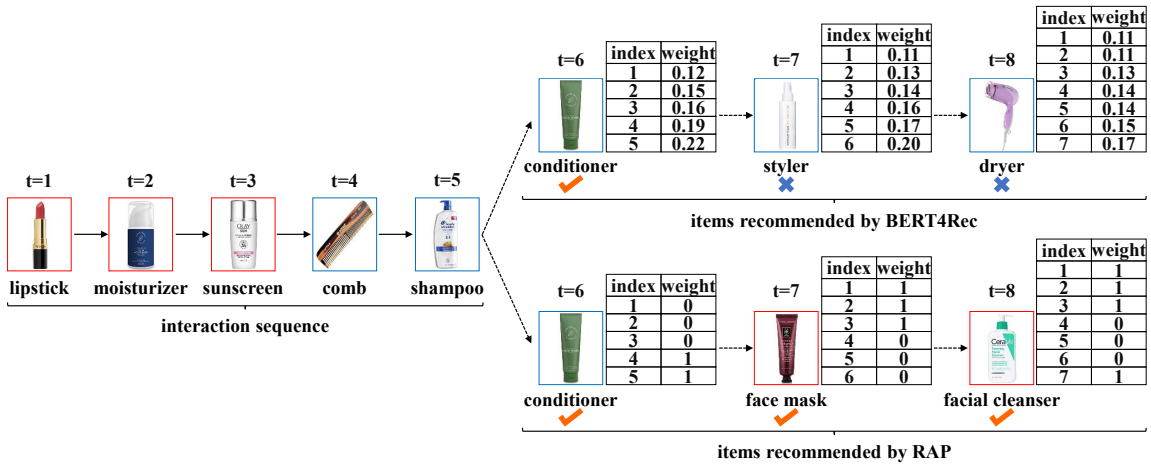

Figure 2: The left row is an interaction sequence of a user sampled from Beauty dataset. The right two rows are recommendation results of BERT4Rec and RAP respectively. Items in red box are face care products, while hair care products are in blue box. For each table, numbers in its left column are index of items, and scores in the right column are the corresponding weights obtained by BERT4Rec and RAP.

5.3 Analysis on RAP

In this section, we conduct experiments on RAP to gain a further insightful understanding. Specifically, we consider three variants as follows:

- RAP^{+p}_{-c}: we retain sequential patterns, and remove the contrastive learning strategy by setting r^- to 0;
- RAP^{-p}_{+c}: we keep the contrastive learning strategy and remove sequential patterns from RAP;
- RAP^{-p}_{-c}: We remove both sequential patterns and the contrastive learning strategy from RAP;

Table 3 reports the performance comparison between RAP and its three sub-models on four datasets. We observe that RAP^{-p}_{-c} obtains the worst performance over all four datasets. It demonstrates making sequence denoising is not an easy task. There is no consistent dominant between RAP^{+p}_{-c}, and RAP^{-p}_{+c}, it indicates both fusing sequential patterns and using contrastive learning process complements each other for better preference learning. Obviously, RAP obtains the best performance against these variants on four datasets.

5.4 Qualitative Analysis

In our model, a major novelty is that we formalize the sequence denoising problem into a MDP. To obtain a better understanding why RAP performs better than other models, shown in Figure 2, we further construct a qualitative analysis with a case study on Beauty dataset. Specifically, we present a snapshot of the interaction sequence for a sampled user, which contains eight items. The first five items are relevant with face care (*lipstick*, *moisturizer* and *sunscreen*) and hair care (*comb* and *shampoo*) respectively. Given the first five items, we use RAP and BERT4Rec to recommend the next three items sequentially. For a convenient demonstration, we only retain the Top-1 recommendation results of two models.

As we can see, both RAP and BERT4Rec recommend the first item *conditioner* correctly, and they both assign higher weights on *comb* and *shampoo*, which are all relevant with *hair care*. However, BERT4Rec fails to make a correct recommendation on the next two items. The reason is that BERT4Rec is based on the MC assumption, hence the last visited item plays a key role. This makes BERT4Rec believe that the user still prefers items related with *hair care*, resulting in higher attention weights on these items (*shampoo*, *conditioner* and *styler*), leading to the wrong recommendation. On the contrary, RAP can well extract the truly sequential dependencies relevant with the target item, and recommends *face mask* and *facial cleanser*, which are relevant with *face care*. That is, RAP can make a more precise preference inference over the sequence dependencies from a more clean subsequence.

6 Conclusion

In this paper, we formulate a sequence denoising problem, and we propose a pattern-enhanced contrastive policy learning network for denoising and recommendation. RAP casts the sequential denoising problem as a form of MDP, and exploits sequential patterns and a contrastive learning process for preference learning. Extensive experiments show that RAP achieves state-of-the-art performance against a series of SOTA solutions.

Right now, we only utilize item interactions to perform denoising. This limited resources would hinder the effective learning of denoising and recommendation. In future, we will choose to extract sequential dependencies from the knowledge graph, which may bring further benefit towards the denoising process.

Acknowledgements

This research work was supported by fundamental Research for the National Natural Science Foundation of China(No.61802029,61872278). We would like to thank for the anonymous reviewers for their valuable comments.

References

[Chen *et al.*, 2018] Yi Chen, Zhuoran Yang, Yuchen Xie, and Zhaoran Wang. Contrastive learning from pairwise measurements. In *NeurIPS*, 2018.

[Cheng *et al.*, 2013] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, 2013.

[Chuang *et al.*, 2020] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *NeurIPS*, 2020.

[Dai and Lin, 2017] Bo Dai and Dahua Lin. Contrastive learning for image captioning. In *NIPS*, 2017.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT, Volume 1*, pages 4171–4186, 2019.

[He and McAuley, 2016] Ruining He and Julian J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, 2016.

[He *et al.*, 2017a] Ruining He, Wang-Cheng Kang, and Julian J. McAuley. Translation-based recommendation. In *RecSys*, 2017.

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

[Hidasi *et al.*, 2016] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[Huang *et al.*, 2018] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*, 2018.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Krichene and Rendle, 2020] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *KDD*, 2020.

[Laskin *et al.*, 2020] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. In *ICML*, volume 119, pages 5639–5650, 2020.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, 2017.

[Lin *et al.*, 2002] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Min. Knowl. Discov.*, 6(1):83–105, 2002.

[Mooney and Roddick, 2013] Carl Mooney and John F. Roddick. Sequential pattern mining - approaches and algorithms. *ACM Comput. Surv.*, 45(2):19:1–19:39, 2013.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidtthieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 2010.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 2019.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 2018.

[Wang *et al.*, 2018] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *AAAI*, 2018.

[Xian *et al.*, 2019] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*, 2019.

[Xin *et al.*, 2020] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. Self-supervised reinforcement learning for recommender systems. In *SIGIR*, 2020.

[Yu *et al.*, 2019] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. Multi-order attentive ranking model for sequential recommendation. In *AAAI*, 2019.

[Zaki, 2001] Mohammed Javeed Zaki. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60, 2001.

[Zhang *et al.*, 2019a] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. Hierarchical reinforcement learning for course recommendation in moocs. In *AAAI*, 2019.

[Zhang *et al.*, 2019b] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, 2019.

[Zhao *et al.*, 2018a] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *RecSys*, 2018.

[Zhao *et al.*, 2018b] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *KDD*, 2018.

[Zhao *et al.*, 2020] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. *arXiv preprint arXiv:2011.01731*, 2020.

[Zheng *et al.*, 2018] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A deep reinforcement learning framework for news recommendation. In *WWW*, 2018.

[Zhou *et al.*, 2020] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *SIGIR*, 2020.

[Zou *et al.*, 2019] Shihao Zou, Zhonghua Li, Mohammad Akbari, Jun Wang, and Peng Zhang. Marlrank: Multi-agent reinforced learning to rank. In *CIKM*, 2019.

[Zou *et al.*, 2020] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. Pseudo dyna-q:A reinforcement learning framework for interactive recommendation. In *WSDM*, 2020.