Personalized Prompt for Sequential Recommendation

Yiqing Wu†, Ruobing Xie†, Yongchun Zhu, Fuzhen Zhuang*, Xu Zhang, Leyu Lin, and Qing He*

Abstract—Pre-training models have shown their power in sequential recommendation. Recently, prompt has been widely explored and verified for tuning after pre-training in NLP, which helps to more effectively and parameter-efficiently extract useful knowledge from pre-training models for downstream tasks, especially in cold-start scenarios. However, it is challenging to bring prompt-tuning from NLP to recommendation, since the tokens of recommendation (i.e., items) are million-level and do not have concrete explainable semantics, and the sequence modeling in recommendation should be personalized. In this work, we first introduce prompt to pre-trained recommendation models and propose a novel Personalized prompt-based recommendation (PPR) framework for cold-start recommendation. Specifically, we build personalized soft prompt via a prompt generator based on user profiles, and enable a sufficient training on prompts via a new prompt-oriented contrastive learning. PPR is effective, parameter-efficient, and universal in various tasks. In both few-shot and zero-shot recommendation tasks, PPR models achieve significant improvements over baselines in three large-scale datasets. We also verify PPR's universality on different pre-trained recommendation models. Finally, we explore and confirm the capability of PPR on other tasks such as cross-domain recommendation and user profile prediction, shedding lights on the promising future directions of better using large-scale pre-trained recommendation models.

Index Terms—recommender system, prompt, pre-training, pre-trained recommendation, contrastive learning

1 INTRODUCTION

PERSONALIZED recommendation aims to provide appropriate items for users according to their preferences, where user historical behavior sequence is an informative source for user understanding. *Sequential recommendation*, which takes users' historical behavior sequences as inputs and outputs the next predicted items, is widely studied and deployed in practice [1], [2], [3], [4]. With the thriving of *pre-training* in NLP [5], there are lots of efforts that bring pre-training into sequential recommendation [6]. These pre-trained recommendation models usually consider user behavior sequences as token sequences in NLP, using pre-training techniques to improve the sequence modeling ability of user behaviors, which can alleviate the sparsity issues in real-world recommendation systems [7].

Recently, with the overwhelming trend of pre-training, how to effectively and efficiently extract useful information from large-scale pre-trained models becomes a promising direction. **Prompt-tuning** [8], [9], [10] is a representative and powerful method that has remarkable superiority over the classical fine-tuning paradigm, especially in zero-shot and few-shot scenarios. It usually inserts hard text templates [8], [11] or soft continuous embeddings [12], [13] as **prompts**, and transforms the downstream tasks into similar well-

trained pre-training tasks. The advantages of prompt-tuning locate in two aspects: (1) it bridges the gap between pre-training and downstream objectives, which could better utilize the knowledge in pre-training models. This advantage will be multiplied in cold-start scenarios. (2) Prompt-tuning only needs to tune a small set of parameters for the prompts and labels, which is more parameter-efficient. Looking back to the sequential recommendation task, the cold-start user issues (including zero-shot and few-shot scenarios) are crucial challenges due to the sparsity of interactions in real-world systems. In this work, with pre-training techniques widely adopted to address cold-start issues, we attempt to bring in prompt-tuning to better extract informative knowledge from the huge pre-trained recommendation models.
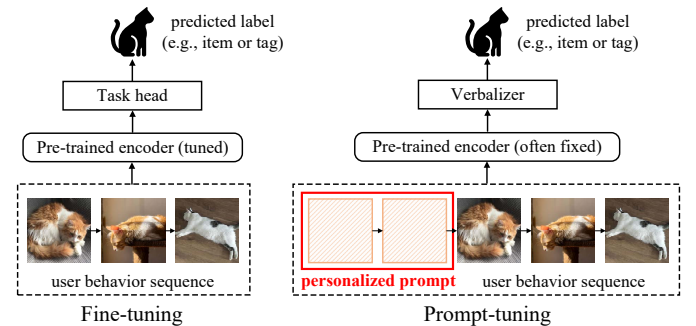

Fig. 1. An example of prompt-tuning in recommendation.

However, adopting prompts in recommendation is non-trivial due to the following challenges: (1) *How to transfer the prompt-tuning paradigm from NLP into recommendation*? Differing from words in NLP, items in recommendation do not have concrete semantic meanings, and thus cannot be directly used to build hard explainable prompts to

• *YiqingWu, Yongchun Zhu and Qing He are with Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China, and University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {wuyiqing20s, zhuyongchun18s, heqing}@ict.ac.cn*
• *Ruobing Xie, Xu Zhang and Leyu Lin are with WeChat, Tencent, China. E-mail: {ruobingxie,xuonezhang,goshawklin}@tencent.com*
• *Fuzhen Zhuang is with Institute of Artificial Intelligence, Beihang University, Beijing 100191, China, and SKLSDE, School of Computer Science, Beihang University, Beijing 100191, China. E-mail: zhuang-fuzhen@buaa.edu.cn*

† *indicates equal contributions.* ∗ *indicates corresponding author.*

prompt pre-trained models. Moreover, it is also challenging to design an appropriate framework to make full use of pre-training knowledge for various recommendation tasks. (2) *How to construct effective personalized prompts for recommendation*? Compared with NLP, recommendation further emphasizes personalization. Hence, the proposed prompts should better be personalized so as to more pertinently extract user-related knowledge from the huge pre-trained recommendation models to downstream tasks.

To address these challenges, we propose a novel **Personalized prompt-based recommendation (PPR)** framework, which first adopts prompt-tuning in pre-trained recommendation models. Specifically, PPR adopts the *personalized soft prefix prompt* learned via our prompt generator based on user information such as user profiles, which elegantly prompts the pre-trained models to provide user-related knowledge for various downstream recommendation tasks within a universal tuning framework. We also propose a *prompt-oriented contrastive learning* via both prompt-based and behavior-based data augmentations to further enhance the training of prompts against data sparsity. Compared with fine-tuning, PPR has the following advantages: (1) PPR enables the prompt-tuning of pre-trained recommendation models, making the usage of pre-training in downstream tasks more effective, personal and parameter-efficient. (2) PPR designs a set of prompt-oriented contrastive learning losses on prompt-enhanced behavior sequences, which enables a more sufficient training for prompts. (3) PPR is effective, parameter-efficient, universal, and easy-to-deploy, which can be conveniently adopted on various pre-trained models and downstream tasks such as cross-domain recommendation and user profile prediction.

In experiments, we conduct extensive evaluations to verify the effectiveness and universality of our personalized prompt-based recommendation. We conduct evaluations on three large-scale datasets in few-shot and zero-shot scenarios, which confirms that PPR can achieve significant improvements in both scenarios. Ablation studies and sparsity analyses are also conducted. Moreover, we also verify PPR's universality on multiple pre-training recommendation models (e.g., SASRec [1] and CL4SRec [4]) and other challenging downstream tasks (e.g., cross-domain recommendation and user profile prediction), shedding light on the promising and wide applications of personalized prompts in practice. The contributions of this work are as follows:

- We propose a novel personalized prompt-based recommendation framework for various recommendation tasks. To the best of our knowledge, we are the first to adopt prompt-tuning for better extracting knowledge from pre-trained recommendation models.
- We design the novel prompt-oriented contrastive learning considering both prompt- and behavior- based augmentations, enabling more sufficient training.
- We have verified the effectiveness and universality of PPR on multiple pre-trained models and tasks including cold-start recommendation, cross-domain recommendation, and user profile prediction. It implies the promising possibility of adopting large-scale pre-trained recommendation models with prompt-tuning as a new recommendation paradigm in the future.

2 RELATED WORKS

Sequential Recommendation. Sequential recommendation models mainly leverage users' chronological behavior sequences to learn users' preferences. Recently, various deep neural networks have been employed for sequence-based recommendation. GRU4Rec [14] proposes to use Gated Recurrent Units in the session-based recommendation. Inspired by the success of Transformer and BERT [5], [15], SASRec [1] and Bert4Rec [2] adopt self-attention mechanisms to model user behavior sequences. The cold-start problem on sequential recommendation also attracts the wide attention of researchers. [16] utilizes a meta-learning mechanism to alleviate cold-start item problems in sequential recommendation. [17] augments short behavior sequence by reversely predicted items. Pre-training & fine-tuning is also a widely-used method for cold-start recommendation [6], [18]. In this work, we focus on improving the pre-training & fine-tuning manner, evaluating PPR with two classical sequential models as the pre-trained models.

Pre-training in Recommendation. Recently, pre-training models have achieved great success in NLP [5], [19] and CV [20], [21]. It aims to learn prior knowledge from general large-scale datasets to help the specific downstream tasks. After pre-training, models are further fine-tuned on downstream supervised signals to fit the specific task. This *pre-training and fine-tuning* paradigm is widely applied to various tasks [5]. With the thriving of a pre-training, many pre-training models have been proposed in recommendation [22], [23], [24]. BERT4Rec [2] adopts masked item prediction in sequential recommendation. S^3Rec [25] pre-trains the sequential encoder considering the correlations among item's attributes, item, subsequence, and sequence. CL4SRec [4] applies contrastive learning (CL) via item crop, mask, and reorder on sequence modeling. PeterRec [26] transfers pre-trained model to solve cross-domain recommendation via Adapters. UPRec [7] further highlights user profiles and social relations in pre-training.

Recently, some works attempt to introduce pre-trained language model (PLM) into recommendation. UniSRec [27] utilizes PLM to learn universal item representations across different domains via items' textual information. PEPLER [28] adopts users'/items' textual attributes as prompts of PLM to generate textual explanations for recommendations. P5 [29] follows PEPLER and designs various prompts to jointly learn from PLM for multiple downstream recommendation tasks. M6-Rec [30] also converts user behaviors and user/item attributes into textual data and models them via PLM. Different from these models that rely on **pre-trained language models** and mainly convert sequential behaviors into texts, our PPR concentrates on the intrinsic **pre-trained recommendation models** and their learned *behavioral* knowledge. Moreover, PPR does not depend on additional textual information and PLMs, and thus is more universal and flexible. Inspired by the successes of pre-training, we propose PPR, which aims to (a) narrow the gap between pre-training and downstream recommendation tasks, and (b) better extract useful personalized knowledge from pre-trained models by replacing fine-tuning with personalized prompt-tuning. We have also deployed PPR on different tasks and pre-training models to verify its
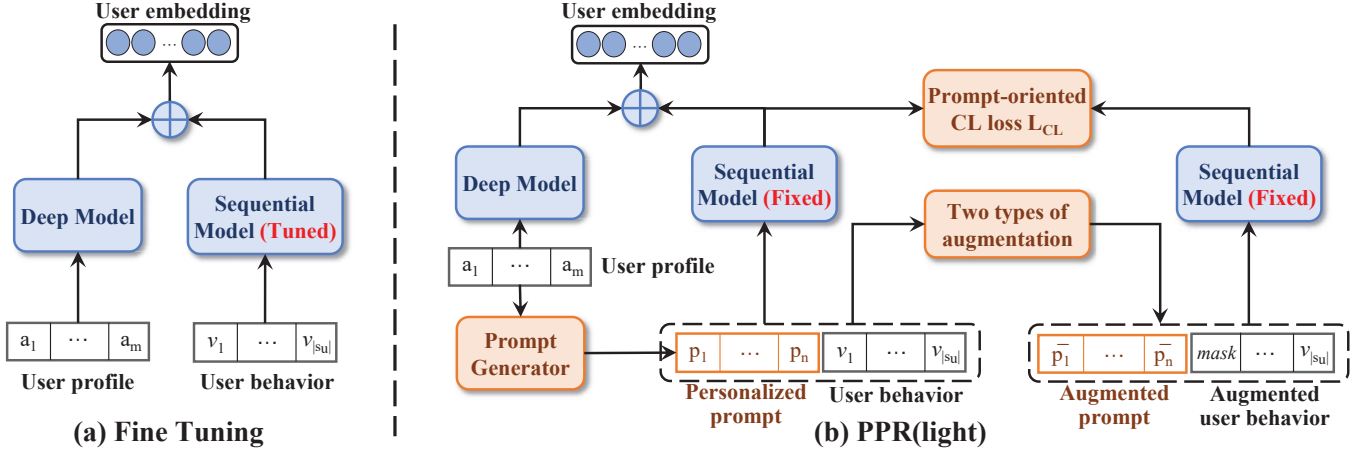
Fig. 2. Overall architecture of (a) the conventional fine-tuning, and (b) our Personalized prompt-based recommendation.

effectiveness.

Prompt Tuning. Prompt-tuning is first proposed in NLP, and is widely explored and dominating, especially in few-shot scenarios [12], [31], [32], [33]. [34] and [8] adopt hard prompts that consist of discrete real words. Considering that manually designing the hard prompt is both time-consuming and trivial, other works [11], [35] focus on automatically searching for hard prompts. In contrast, soft prompts are composed of several continuous learnable embeddings randomly initialized. Prefix-tuning [12] optimizes continuous prompts for generation tasks in NLP. [36] adopts prefix prompts for selective fairness. PPT [37] further conducts pre-training on the prompts to better initialize soft prompts. In PPR, we first introduce prompts to better utilize pre-trained recommendation models. Different from prompts in NLP, we design personalized prompts.

3 METHODOLOGY

3.1 Preliminaries

3.1.1 Background of Prompt in NLP

In NLP pre-training, prompts are often a piece of natural language inserted into the input text sequence. For example, in sentiment analysis, a prompt "it is [mask]" is inserted after the review "a real joy" as a natural sentence: "a real joy, it is [mask]". In prompt-tuning, the predicted tokens of "[mask]" (e.g., great) will be mapped to the sentiment labels (e.g., positive) via a task-specific verbalizer. In this case, the original task can be formulated as a masked language model task, which has been fully optimized in pre-training and thus has better performance [8], [31], [38]. The advantages of prompt-tuning are as follows: (1) it can better extract useful knowledge in pre-trained models via prompts based on similar modeling and objectives fully learned in pre-training, especially for few-shot scenarios compared to fine-tuning [37], and (2) it is more parameter-efficient compared to fine-tuning, for the tuned parameters in prompt-tuning are far fewer than those of pre-trained models.

3.1.2 Goals of PPR

In this work, we propose Personalized prompt-based recommendation to provide a more effective and parameter-efficient tuning for (cold-start) downstream tasks based on

pre-trained sequential recommendation models. The *pre-training and fine-tuning* is a classical and widely-used training paradigm, which first trains a pre-trained model on a large-scale general dataset, and then fine-tunes the whole pre-trained model by the supervised information of downstream tasks. Recently, the **pre-training and prompt-tuning** paradigm has been widely verified in NLP as in Sec. 2, where the prompts and verbalizer are mainly updated for efficiency. PPR attempts to better extract useful personalized information from pre-trained models via prompt-tuning rather than fine-tuning, and thus fine-tuning is the most essential baseline. Precisely, the main task that our PPR focuses on is cold-start recommendation (since prompt-tuning functions well in few-shot learning), which consists of both *few-shot recommendation* and *zero-shot recommendation*. Moreover, we also deploy PPR for other downstream tasks such as *cross-domain recommendation* and *user profile prediction* in Sec. 4.9 to analyze PPR's universality.

3.1.3 Notions of PPR

The key notions of PPR are defined as follows. We denote user and item as $u \in U$ and $v \in V$. U and V are the overall user and item sets. Each user u has a historical behavior sequence $s_u = \{v_1^u, v_2^u, ..., v_{|s_u|}^u\}$ (of length $|s_u|$) ordered by time. Each user has m attributes $A_u = \{a_1^u, a_2^u, ..., a_m^u\}$ (i.e., user profiles such as age and gender). In cold-start recommendation, we split users into a warm user set U^W (for pre-training) and a cold user set U^C (for tuning and evaluation) by their behavior sequence length. Then, we pre-train a sequential model $f_{seq}(\{v_1^u, v_2^u, ..., v_{|s_u|}^u\}|\Theta)$ to learn user representations by warm users' historical behaviors, where Θ is the pre-trained model's parameters. After that, we fine-tune the pre-trained model on cold users' historical behaviors and get the fine-tuned model $f_{seq}(\{v_1^u, v_2^u, ..., v_{|s_u|}^u\}|\hat{\Theta})$, where $\hat{\Theta}$ is the fine-tuned model's parameters. Different from fine-tuning, our PPR inserts personalized prompts (i.e., prefix tokens) before behavior sequences, noted as $f_{seq}(\{p_1^u, p_2^u, ..., p_n^u, v_1^u, v_2^u, ..., v_{|s_u|}^u\}|\Theta, \vartheta)$, where $\{p_1^u, p_2^u, .., p_n^u\}$ is the personalized prompt built from user profiles, and ϑ is the prompt generator's parameters.

3.2 Overall Framework

The overall framework of PPR is illustrated in Fig. 2. For each user, we first build the personalized prompts according to user profiles via the prompt generator, and insert them into the beginning of the user behavior sequence. The prompt-enhanced sequence is then fed into the pre-trained sequential model to generate user's behavioral representations. Moreover, besides the prompt generator, the user profiles are further fed into another Deep model to generate user's attribute-level representations. Finally, the user's behavioral and attribute-level preferences are combined to get the final user representation. To enable a more sufficient training of personalized prompts, we further design a prompt-oriented contrastive learning loss, adopting two augmentations on both the prompt generator and sequence modeling parts. We design two PPR versions for different practical demands. In PPR(light), the pre-trained model is fixed during tuning as classical prompt-tuning in NLP, which is more parameter-efficient. While in PPR(full), all prompts and pre-trained parameters are tuned specially for the characteristics of recommendation.

3.3 Pre-training of PPR

We first introduce the pre-training part of PPR. There are various self-attention based sequential models [1], [2], [4] verified to be effective as pre-training models. Following [4], we also use the classical SASRec [1] as our pre-training model. Specifically, SASRec stacks Transformer$(\cdot)$ blocks to encode the historical behavior sequence. For the input behavior sequence s_u, we define its l-layer's behavior matrix as $\boldsymbol{H}_u^l = \{\boldsymbol{h}_{u,1}^l, \boldsymbol{h}_{u,2}^l, \cdots, \boldsymbol{h}_{u,|s_u|}^l\}$, where $\boldsymbol{h}_{u,i}^l$ is the i-th behavior's representation of u at the l-th layer. The $(l+1)$-layer's behavior matrix $\boldsymbol{H}_u^{l+1}$ is then learned as follows:

$$\begin{aligned} \boldsymbol{H}_u^{l+1} &= \text{Transformer}^l(\boldsymbol{H}_u^l), \\ \boldsymbol{u}_o &= f_{seq}(s_u|\Theta) = \boldsymbol{h}_{u,|s_u|}^L. \end{aligned} \quad (1)$$

$\boldsymbol{u}_o$ is the final user representation of u learned in pre-training to predict the user's next items, which is generated as the last behavior's representation at the L-th layer (i.e., $\boldsymbol{h}_{u,|s_u|}^L$). Here, L is the number of Transformer layers.

Following classical ranking models [1], [4], [39], the pre-training model is optimized under the objective L_o as:

$$L_o = - \sum_{(u,v_i)\in S_+^w} \sum_{(u,v_j)\in S_-^w} \log \sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_i - \boldsymbol{u}_o^\top \boldsymbol{v}_j), \quad u \in U^w. \quad (2)$$

$(u, v_i) \in S_+^w$ indicates the positive set where u has clicked v_i, and $(u, v_j) \in S_-^w$ indicates the negative set where v_j is randomly sampled negative items. $\sigma(\cdot)$ is the sigmoid function. We optimize the parameters Θ of the pre-training model via L_o on the pre-training dataset of $u \in U^w$. Differing from classical prompt-tuning in NLP, which usually adopts the masked language model (MLM) pre-training task [5], PPR mainly concentrates on the next-item prediction task in pre-training. It is because the next-item prediction task perfectly fits our downstream tasks (cold-start recommendation and user profile prediction) with PPR, which can stimulate the maximum potential of prompt. Note that our PPR can be flexibly adopted on different pre-training models. We verify PPR's universality on CL4SRec [4] in Sec. 4.7.

3.4 Personalized Prompt-tuning

After pre-training, PPR conducts a personalized prompt-tuning instead of fine-tuning to better extract useful information from pre-trained models to downstream tasks.

3.4.1 Personalized Prompt Generator

The key of our PPR is generating an effective prompt that helps to narrow the gap between pre-trained models and downstream tasks. However, it is challenging to find appropriate prompts in recommendation, since (1) it is difficult to build hard prompts (i.e., some real tokens) in PPR. Unlike words in NLP, the tokens (i.e., items) in recommendation do not have explicit meaningful semantics. (2) Moreover, unlike NLP, recommendation should be personalized. thus the prompts should also be customized for different users. In a sense, each user's recommendation can be viewed as a task, while there are millions of users in a real-world system. It is impossible to design prompts manually for each user.

In PPR, to automatically and effectively build personalized prompts for all users, we rely on essential and informative user profiles. *User profiles* can be produced by all types of user information, such as user static attributes (e.g., age, gender, location), user cumulative interests, and user behaviors in other domains. In the task of cold-start recommendation, we mainly consider the user static attributes A_u to build prompts. Specifically, we concatenate m user profile embeddings as $\boldsymbol{x}_u = [\boldsymbol{a}_1^u||\boldsymbol{a}_2^u||\cdots||\boldsymbol{a}_m^u]$, where $\boldsymbol{a}_i^u$ is the i-th user profile embedding. We conduct a Multi-layer perceptron (MLP) to learn the personalized prompt representation $\boldsymbol{P}^u = \{\boldsymbol{p}_1^u, \cdots, \boldsymbol{p}_n^u\}$ containing n tokens as:

$$\boldsymbol{P}^u = \text{PPG}(\boldsymbol{x}_u|\vartheta) = \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{x}_u + \boldsymbol{b}_1) + \boldsymbol{b}_2, \quad (3)$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d_1 \times d'}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d' \times d_2 \times n}$, $\boldsymbol{b}_1 \in \mathbb{R}^{d'}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{d_2 \times n}$ are trainable parameters in ϑ. d_1, d', and d_2 are the embedding sizes of concatenated user profiles $\boldsymbol{x}_u$, hidden layer, and output prompts respectively. n is the number of prompt tokens. We choose the current $\text{PPG}(\cdot)$ function since it is simple and effective. It is also convenient to adopt other neural networks to build the prompts $\boldsymbol{P}^u$ from $\boldsymbol{x}_u$.

Inspired by the success of prefix-tuning [12], we adopt the prompt tokens as a prefix and connect them with user behavior sequence as: $\hat{s}_u = \{p_1^u, p_2^u, ..., p_n^u, v_1^u, v_2^u, ..., v_{|s_u|}^u\}$. This *prompt-enhanced sequence* contains both task-specific and user-specific information. For example, if $A_u = \{20\text{-}$ year-old, female$\}$, the prompt-enhanced sequence $\hat{s}_u$ can be translated as "In cold-start recommendation, a user is a 20-year-old female, she likes $v_1, v_2, \cdots$ ", which provides a more personalized context of user behavior sequences that better prompts the sequential model to give appropriate recommendations. Different from the side Deep model of Fine-tuning in Fig. 2(a) that captures user profile information separately from user behaviors, our PPR naturally fuses heterogeneous user profiles into user historical behaviors, and models them via a joint sequential modeling inherited from the pre-trained recommendation model. In this case, the user-related knowledge hidden in pre-trained recommendation models can be fully activated for downstream tasks, which is essential in cold-start scenarios.

3.4.2 Prompt-tuning of PPR

After generating the personalized prompts, we need to construct the final user representation for recommendation. Precisely, we first input the prompt-enhanced sequence $\hat{s}_u$ to the pre-trained sequential model in Eq. (1) to get the user behavioral preference $\boldsymbol{u}_s$. Similar to fine-tuning in Fig. 2, we also use a Deep model (MLP) to learn the user attribute preference $\boldsymbol{u}_a$ from user profiles. Finally, both $\boldsymbol{u}_s$ and $\boldsymbol{u}_a$ are combined to get the final user representation $\boldsymbol{u}_p$. We have:

$$\boldsymbol{u}_p = \boldsymbol{u}_a + \boldsymbol{u}_s, \quad \boldsymbol{u}_a = \mathrm{MLP}_a(\boldsymbol{x}_u|\phi), \quad \boldsymbol{u}_s = f_{seq}(\hat{s}_u|\Theta, \vartheta). \tag{4}$$

Θ, ϑ, and ϕ are parameters of the pre-trained sequential model, the prompt generator, and the Deep model for $\boldsymbol{u}_a$, respectively. Finally, we follow the pre-training objective in Eq. (2) to build the optimization objective L_p of our prompt-tuning for col-start recommendation as follows:

$$L_p = -\sum_{(u,v_i)\in S_+^c} \sum_{(u,v_j)\in S_-^c} \log \sigma(\boldsymbol{u}_p^\top \boldsymbol{v}_i - \boldsymbol{u}_p^\top \boldsymbol{v}_j), \quad u \in U^c. \tag{5}$$

S_+^c and S_-^c are the same positive and negative sample sets in fine-tuning. We adopt similar training objectives for pre-training and prompt-tuning, aiming to better use pre-trained knowledge for downstream tasks via parameter-efficient prompt-tuning.

To tune these parameters, we propose two prompt tuning strategies, PPR(light) and PPR(full), to balance effectiveness and efficiency for practical demands.

PPR(light). In PPR(light), we merely update the newly-introduced parameters, i.e., ϑ of the prompt generator in Eq. (3) and ϕ of the user profile learner in Eq. (4), with other parameters fixed. It is similar to the typical prompt-tuning manner in NLP [12]. This is a straightforward and efficient prompt-tuning manner, which completely relies on the pre-trained recommendation model in sequence modeling and item representation learning. The tuned parameters of PPR(light) are greatly reduced compared to fine-tuning (note that even the item embeddings to be predicted are also fixed for efficiency). The tuned personalized prompts work as an inducer, which smartly extracts useful knowledge related to the current user from a large-scale pre-trained model.

PPR(full). However, due to the huge gaps between NLP and recommendation tasks, the widely-verified original "light" prompt-tuning does not always perform well enough in downstream tasks. It is because that (a) plenty of NLP downstream tasks (e.g., sentiment analysis, text classification) only have limited numbers of predicted labels, and (b) high-frequent words are often fewer than several thousand. On the contrary, items (also labels to predict) in recommendation are often million-level in practice, while they are fixed in PPR(light). Hence, we propose another PPR(full) manner for more comprehensive tuning. Specially for a recommendation, which further tunes the pre-trained model's parameters Θ (including parameters of item embeddings and sequential model) besides ϑ and ϕ. In a sense, PPR(full) can be regarded as prompt-enhanced tuning by combining fine-tuning and PPR(light).

3.5 Prompt-oriented Contrastive Learning

The main challenge of cold-start recommendation is the lack of sufficient tuning instances, and similar issues also exist in prompt-tuning. Recently, contrastive learning (CL) has shown its power in recommendation [4], [25]. These CL-based models usually conduct self-supervised learning (SSL) as supplements to supervised information via certain data augmentations, which could obtain more effective and robust user representations and alleviate the data sparsity issues. Inspired by those methods, we also adopt CL as auxiliary losses via two types of data augmentations based on our prompt-enhanced sequences.

3.5.1 Prompt-based Augmentation

In real-world systems, users' attributes are usually noisy or even missing, while they are the main source of our personalized prompts, whose qualities are essential, especially in zero-shot scenarios. Therefore, we design a prompt-based data augmentation to improve the effectiveness and robustness of the prompt generator. Specifically, we conduct random element-level masking on the feature elements of user profile embeddings $\boldsymbol{x}_u$ to obtain $\bar{\boldsymbol{x}}_u$ with a certain mask ratio γ_1. Formally, the augmented prompt-enhanced behavior sequence is noted as $\bar{s}_u^1 = \{\bar{p}_1^u, \bar{p}_2^u, ..., \bar{p}_n^u, v_1^u, v_2^u, ..., v_{|s_u|}^u\}$, where $\bar{\boldsymbol{P}}^u = \mathrm{PPG}(\bar{\boldsymbol{x}}_u|\vartheta)$ is learned from masked attributes. This augmentation can also avoid the possible overfitting of our prompt generator on some abnormal profiles.

3.5.2 Behavior-based Augmentation

Besides the prompt-based augmentation, we also conduct data augmentations on the original user historical behaviors in the prompt-enhanced sequences. Following previous CL-based models [4], [25], we randomly zero-mask proportional items in user behavior sequence with the mask ratio γ_2. Intuitively, the augmented prompt-enhanced sequence is noted as $\bar{s}_u^2 = \{p_1^u, p_2^u, ..., p_n^u, v_1^u, [mask], ..., v_{|s_u|}^u\}$, strengthening the sequence modeling ability from another aspect. These two augmentations cooperate well in SSL.

3.5.3 Contrastive Learning

In prompt-oriented contrastive learning, we hope PPR can distinguish whether two user representations learned from (augmented) prompt-enhanced behavior sequences derive from the same user. To achieve this goal, we need to minimize the differences between the original and augmented sequences of the same users while maximizing the gaps between different users' representations. Specifically, for a batch B with size N, we apply the above two augmentations to each user u, and get the augmented sequences $\bar{s}_u^1$ and $\bar{s}_u^2$. We regard the original and the corresponding augmented user behavioral representations of u as the positive pair $(\boldsymbol{u}_s, \bar{\boldsymbol{u}}_s)$. The rest augmented user representations $\bar{\boldsymbol{u}}_s'$ of other users u' in the batch form the negative set S_-^u of u. We use the cosine similarity $\mathrm{sim}(\cdot, \cdot)$ to measure the similarity. Formally, the final CL loss function L_{CL} is formulated as:

$$L_{CL} = -\sum_B \sum_{u \in U^c} \log \frac{\exp(\mathrm{sim}(\boldsymbol{u}_s, \bar{\boldsymbol{u}}_s)/\tau)}{\sum_{u' \in B} \exp(\mathrm{sim}(\boldsymbol{u}_s, \bar{\boldsymbol{u}}_s')/\tau)}. \tag{6}$$

Here, the augmented user representation $\bar{\boldsymbol{u}}_s$ has two forms $f_{seq}(\bar{s}_u^1|\Theta, \vartheta)$ or $f_{seq}(\bar{s}_u^2|\Theta, \vartheta)$. τ is the temperature hyper-parameter. The prompt-oriented contrastive learning loss is

used as an auxiliary task of L_p in prompt-tuning to train the personalized prompts fully. The overall loss L_{all} is defined with the loss weight λ as follows:

$$L_{all} = L_p + \lambda L_{CL}. \tag{7}$$

3.6 Model Designs and Complexity

Model Designs. In NLP, pre-training with prompt performs better than classical fine-tuning especially in few-shot scenarios. Recently, pre-training has also shown its power in recommendation [2], [36]. Hence, we attempt to use prompt-tuning to help pre-trained recommendation models to *more effectively and parameter-efficiently fit downstream tasks* in recommendation. The advantages of prompts and using prompts in our tasks are listed in Sec. 1 and Sec. 3.

We adopt user profiles to build the personalized prompts in tuning, since user profiles can prompt the pre-trained model to give more prior knowledge *related to the current user* in sequential modeling, and they are naturally available and widely-used in practical systems. We restrict PPR to merely utilize user profiles in the downstream tuning for universality and better user privacy protection in potential future applications. We do not modify the pre-training part, since we want the proposed PPR to be suitable for different potential large-scale pre-trained recommendation models. Moreover, we design two versions PPR(light) and PPR(full). It is because that different from NLP, the tokens (i.e., items) in recommendation are usually more than a million levels. Hence, the item embeddings appearing in the downstream tasks should be better tuned (just like the verbalizer tuned in prompt-tuning of NLP). Nevertheless, PPR(light) still achieves good performance. We could choose PPR versions flexibly according to practical demands.

Complexity. PPR(light) is parameter-efficient, since it only needs to tune and store a few parameters compared with fine-tuning (i.e., Θ is fixed). In our datasets, the numbers of parameters to be updated in PPR(light) are merely 0.3%, 0.25%, and 0.6% of those in fine-tuning, respectively. PPR(full) is designed specifically for better representation learning of million-level items in recommendation. It has comparable updated parameters with better performances. Table 1 shows the details statistics. In both tuning and serving, Fine-tuning and PPR models have similar time costs.

TABLE 1
Detailed statistics of tuned parameters of three datasets.

Dataset	fine-tuning	PPR(light)	PPR(full)
CIKM	100%	0.3%	100.3%
QQBrowser	100%	0.25%	100.2%
AliEC	100%	0.6%	100.6%

4 EXPERIMENTS

In this section, we conduct extensive experiments to answer the following six research questions: (**RQ1**): How does PPR perform in few-shot scenarios (Sec. 4.4)? (**RQ2**): Can PPR work well in the challenging zero-shot recommendation

(Sec. 4.5)? (**RQ3**): What are the effects of different components in PPR (Sec. 4.6)? (**RQ4**): Can PPR also work on different pre-training models (Sec. 4.7)? (**RQ5**): Can PPR achieve improvements under different data sparsity (Sec. 4.8)? (**RQ6**): Is PPR still effective to be adopted on other downstream recommendation tasks (Sec. 4.9)?

4.1 Datasets

We evaluate PPR on three real-world open datasets, namely CIKM, QQBrowser, and AliEC&AliAD. In all datasets, the users are split into warm users and cold-start users according to a threshold of interacted items (i.e., users having less than 10 clicks are empirically viewed as cold-start users [40], [41]). In real-world systems, the behavioral patterns and recommendation models of cold-start users are usually different from those of warm users. Hence, we regard the general recommendation (pre-training) and the cold-start recommendation as two tasks. The click instances of warm users are used as the pre-train set, while those of cold-start users are used as the tuning/test set for downstream tasks. For cold-start user recommendation, we randomly split the cold-start users into train (in tuning) (80%) and test (20%) sets following [40]. More details are in Table 2.

TABLE 2
Detailed statistics of three datasets.

Dataset	# user	# item	# pre-training instance	# tuning instance
CIKM	80,964	87,894	2,103,610	143,726
QQBrowser	134,931	97,904	15,359,880	263,572
AliEC	104,984	109,938	8,768,915	39,292

CIKM. The CIKM dataset is an E-commerce recommendation dataset released by Alibaba[1]. It has 60 thousand warm users with 2.1 million click instances in the pre-train set. Other 21 thousand cold-start users with 143 thousand instances are used for tuning and testing. Each user has 3 attributes: gender, age, and consumption level.

QQBrowser. This dataset is collected from QQ Browser [26] on news/videos. It has 107 thousand warm users and 28 thousand cold users. Each user has 3 attributes: gender, age, and life status.

AliEC&AliAD. This dataset contains two sub-datasets: AliEC for E-commerce and AliAD for advertising[2]. AliAD is much sparser than AliEC. For cold-start recommendation and user profile prediction tasks, we evaluate on AliEC. It has nearly 99 thousand warm users and 6.4 thousand cold users. For cross-domain recommendation, we set AliEC as the source domain and AliAD as the target domain. Each user has 8 attributes: user SEGID, user group ID, gender, age, consumption level, shopping level, occupation, and city.

4.2 Competitors

In this work, we adopt the representative SASRec [1] as our base pre-training model, while it is also convenient to

1. https://tianchi.aliyun.com/competition/entrance/231719/introduction
2. https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

TABLE 3
Results on few-shot recommendation. All improvements are significant over baselines (t-test with p < 0.05).

Dataset	Model	AUC	HIT@5	NDCG@5	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
CIKM	BERT4Rec	0.8482	0.5191	0.4279	0.6235	0.4616	0.7367	0.4902	0.8965	0.5220
	SASRec	0.8676	0.5808	0.4924	0.6722	0.5220	0.7717	0.5471	0.9100	0.5746
	MeLU	0.8255	0.4647	0.3788	0.5692	0.4124	0.6963	0.4444	0.8773	0.4803
	CL4SRec	0.8676	0.5840	0.4930	0.6737	0.5252	0.7726	0.5469	0.9099	0.5742
	pre-train	0.8630	0.5779	0.4906	0.6695	0.5203	0.7660	0.5446	0.9039	0.5721
	fine-tuning	0.8731	0.5886	0.4948	0.6836	0.5255	0.7837	0.5508	0.9159	0.5772
	PPR(light)	**0.8780**	0.5895	0.4955	0.6854	0.5265	0.7894	0.5528	**0.9230**	0.5795
	PPR(full)	0.8774	**0.5918**	**0.4976**	**0.6889**	**0.5290**	**0.7903**	**0.5547**	0.9211	**0.5807**
QQBrowser	BERT4Rec	0.9546	0.7782	0.6295	0.8734	0.6606	0.9386	0.6772	0.9858	0.6867
	SASRec	0.9629	0.8077	0.6609	0.8974	0.6902	0.9543	0.7048	0.9904	0.7121
	MeLU	0.9318	0.6616	0.5020	0.7929	0.5447	0.8981	0.5714	0.9810	0.5882
	CL4SRec	0.9626	0.8064	0.6543	0.8958	0.6864	0.9550	0.7015	0.9898	0.7078
	pre-train	0.9572	0.7842	0.6338	0.8818	0.6657	0.9448	0.6817	0.9873	0.6904
	fine-tuning	0.9645	0.8142	0.6659	0.9017	0.6944	0.9578	0.7087	0.9919	0.7156
	PPR(light)	0.9640	0.8068	0.6545	0.8982	0.6843	0.9575	0.6994	**0.9926**	0.7066
	PPR(full)	**0.9652**	**0.8169**	**0.6680**	**0.9031**	**0.6960**	**0.9589**	**0.7104**	0.9920	**0.7171**
AliEC	BERT4Rec	0.8758	0.5543	0.4370	0.6738	0.4757	0.7898	0.505	0.9266	0.5324
	SASRec	0.8842	0.5884	0.4663	0.6997	0.5024	0.8083	0.5299	0.9316	0.5546
	MeLU	0.8900	0.5759	0.4512	0.6955	0.4899	0.8131	0.5197	0.9326	0.5457
	CL4SRec	0.8900	0.6036	0.4778	0.7137	0.5134	0.8239	0.5413	0.9379	0.5641
	pre-train	0.8838	0.5880	0.4710	0.7006	0.5073	0.8110	0.5351	0.9308	0.5592
	fine-tuing	0.8907	0.6058	0.4851	0.7189	0.5217	0.8212	0.5475	0.9371	0.5708
	PPR(light)	**0.8975**	0.6123	0.4873	**0.7275**	0.5246	**0.8363**	0.5521	**0.9422**	0.5734
	PPR(full)	0.8941	**0.6126**	**0.4896**	0.7241	**0.5256**	0.8284	**0.5522**	0.9374	**0.5739**

deploy PPR on other pre-trained recommendation models (e.g., CL4SRec [4]). In few-shot recommendation. We compare PPR with several competitive models as follows:

(1) **SASRec** [1], which is a classical sequential recommendation model that introduces self-attention in behavior modeling. It is optimized by both pre-train and tuning sets. It is used as PPR's pre-trained model.

(2) **BERT4Rec** [2], which is a widely-used pre-training recommendation model based on BERT. It uses masked item prediction as its pre-training task.

(3) **MeLU** [40], which is a typical meta-learning based cold-start recommendation model. It views new users as new tasks and adopts MAML for fast adaptations.

(4) **CL4SRec** [4], which is one of the SOTA CL-based sequential recommendation models also based on SASRec. It alleviates the sparsity and cold-start issues with several sequence-based augmentations and contrastive learning. We also deploy our PPR framework with CL4SRec in Sec. 4.7.

(5) **Pre-train**. We directly conduct the pre-trained model (SASRec) on the test set as a baseline for comparison.

(6) **Fine-tuning**. It is a widely-used and strong tuning method to utilize pre-trained models for downstream tasks. It tunes all pre-trained model's parameters in tuning. Note that the fine-tuning model also jointly considers the same user profiles and historical behaviors as in Fig. 2.

We also implement two PPR versions, namely PPR(light) and PPR(full) as introduced in Sec. 3.4.2. PPR(light) only updates the parameters ϑ of the prompt generator and ϕ of the user profile learner with other parameters fixed, which is more parameter-efficient compared to fine-tuning for ϑ and ϕ is far smaller than the pre-trained model's parameters Θ (less than 1% in Table 1). PPR(full) updates all parameters as fine-tuning. We should highlight that all models share

the **same input features** (including *user profiles* encoded by the Deep model and all behaviors) and training instances in tuning for fair comparisons (except for pre-training that only uses the behavioral information of the pre-train set). Fine-tuning and PPR have the same pre-trained model.

4.3 Experimental Settings

Parameter Settings. For fair comparisons, the embedding size is 64 and batch size is 256 equally for all methods. We optimize all models by Adam. We conduct a grid search for hyper-parameters. We search models' learning rates among $\{1e-3, 3e-4, 1e-4, 3e-5, 1e-5, 3e-6, 1e-6\}$. The best learning rates are $1e-5$ for fine-tuning and PPR(full), and $1e-4$ for the rest models. In PPR, the prompts are generated via Eq. (3) taking user profiles as input features, where MLP is verified strong enough. For efficiency, the prompt length is set as 1 (we find that longer prompts do not bring in much improvement). The mask ratios of two data augmentations are 0.2 and the loss weight $\lambda = 0.1$.

Evaluation Protocols. We mainly focus on the few-shot and zero-shot recommendation tasks of cold-start users. For all cold-start users in the test set, their first clicked items are used for zero-shot recommendation (since there is no historical behavior at this time), while the rest click behaviors are used for few-shot recommendation. We use the classical AUC, top-N hit rate (HIT@N), and Normalized Discounted Cumulative Gain (NDCG@N) as our evaluation metrics. For HIT@N and NDCG@N, we report top 5, 10, 20 and 50. For each ground truth, we randomly sample 99 items that the user did not click as negative samples [2], [25].

TABLE 4
Results on zero-shot recommendation. The improvements are significant over baselines (t-test with p<0.05).

Dateset	Model	AUC	HIT@5	NDCG@5	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
CIKM	fine-tuning	0.7753	0.2991	0.2061	0.4235	0.2463	0.5946	0.2894	0.8460	0.3394
	PPR(light)	**0.7845**	0.2998	0.2063	**0.4337**	0.2493	**0.6041**	**0.2923**	**0.8621**	0.3436
	PPR(full)	0.7825	**0.3069**	**0.2100**	0.4329	**0.2507**	0.5974	0.2920	0.8616	**0.3446**
QQBrowser	fine-tuning	0.8743	0.4950	0.3538	0.6427	0.4016	**0.7822**	0.4368	0.9393	0.4684
	PPR(light)	0.8635	0.4829	0.3418	0.6337	0.3907	0.7770	0.4269	0.9262	0.4569
	PPR(full)	**0.8757**	**0.5012**	**0.3612**	**0.6443**	**0.4074**	0.7813	**0.4421**	**0.9411**	**0.4742**
AliEC	fine-tuning	0.8377	0.3905	0.2713	0.5675	**0.3286**	0.7252	0.3686	0.9006	**0.4039**
	PPR(light)	0.8349	**0.4014**	**0.2743**	0.5575	0.3247	**0.7337**	**0.3693**	0.9022	0.4028
	PPR(full)	**0.8380**	0.3983	0.2708	**0.5691**	0.3256	0.7283	0.3661	**0.9037**	0.4013

4.4 Few-shot Recommendation (RQ1)

We first evaluate models on the few-shot recommendation. Table 3 shows the results on three datasets. We can find that:

(1) Our personalized prompt-based model outperforms all baselines on all metrics in three datasets (the significance level is $p < 0.05$ via t-test). It indicates that our personalized prompts can better extract useful information related to the current user from the huge knowledgeable pre-training models, which is beneficial especially for cold-start scenarios. We should highlight that fine-tuning is a very strong baseline (some prompt-tuning models in NLP only have comparable or even worse performance compared to fine-tuning). Nevertheless, PPR models still outperform fine-tuning and other classical cold-start recommendation models, including MeLU and CL4SRec (note that these baselines also have the same *user profiles* as inputs). The advantages of PPR mainly derive from that (a) PPR conducts a unified well-learned sequential modeling on both user profiles and few-shot behaviors, and (b) user profiles are considered to enable a personalized user behavior modeling. Both factors are beneficial especially in cold-start scenarios.

(2) Comparing among different PPR settings, we observe that PPR(full) generally achieves better results in few-shot scenarios. Different from NLP tasks such as sentiment analysis with several labels, the label set of recommendation is far larger (often million-level). Moreover, items in recommendation do not have concrete semantics as words. Hence, the current pre-trained recommendation models cannot perfectly learn all item representations, and tuning is still beneficial. It is natural that the tuned item representations of PPR(full) can further improve the performances with comparable time and memory costs of fine-tuning.

(3) PPR(light) achieves some SOTA results on HIT@N in AliEC dataset, and generally outperforms fine-tuning in two datasets. It is challenging since PPR(light) only tunes the personalized prompt part with all pre-trained parameters unchanged. The effectiveness and efficiency improvements of PPR(light) are encouraging as other prompts in NLP [12], [33]. PPR(light) outperforms PPR(full) in a sparser AliEC dataset, which confirms that it is difficult to tune all large-scale pre-trained parameters with extremely sparse downstream instances. It emphasizes the advantages of PPR(light) compared to fine-tuning and PPR(full) besides efficiency. We can select between the full and light PPR versions according to the practical priority of effectiveness and efficiency.

4.5 Zero-shot Recommendation (RQ2)

The data sparsity issue is extremely serious, where zero-shot users widely exist in practical systems. Most conventional sequential recommendation models (including SASRec, MeLU, and CL4SRec) cannot handle the zero-shot scenarios, since there is no historical behavior. We attempt to jointly address the zero-shot recommendation with the same PPR framework. For PPR, we directly input user profiles into the prompt generator as the few-shot recommendation without behavioral inputs. For fine-tuning baseline, only the user profile learner (i.e., the MLP in Eq. (4)) is activated to learn from user profiles. From Table 4 we can observe that:

(1) Our PPR models still achieve the best performances on most metrics of three datasets, which confirms the effectiveness of PPR in zero-shot user understanding via fully reusing pre-training knowledge. We should highlight that getting improvements in zero-shot scenarios is challenging, since all models share the same sparse training instances and user profiles in tuning. Compared with fine-tuning, PPR makes full use of pre-trained models to encode user profiles, which is the main reason for our improvements.

(2) Currently, the user profiles for prompt construction are not that sufficient in the open datasets, which may limit the modeling ability of prompt-tuning. It can be expected that the improvements will be more significant if enhanced with more user features. We have also evaluated the effectiveness of other prompt input features and construction methods (e.g., built via user behavioral embeddings learned from other domains) for other tasks in Sec. 4.9.

Joint Few-shot and Zero-shot Recommendation. Table 3 and 4 demonstrate the effectiveness of our PPR in both few-shot and zero-shot recommendation. Considering the storage efficiency and maintenance cost in online deployment, we further conduct a challenging evaluation, using one set of model parameters to jointly address two cold-start recommendation tasks. Table 5 shows the results of these joint cold-start recommendation task. We find that: PPR models significantly outperform fine-tuning by a larger margin in this joint scenarios. It indicates that the powerful personalized prompts could help to understand both few-shot and zero-shot users simultaneously in practice, which cannot be accomplished well by most classical sequential models as long as they rely on user behaviors.

TABLE 5
Results of joint cold-start recommendation (few-shot+zero-shot). Improvements are significant (t-test with p<0.05).

| Database	Model	AUC	HIT@5	NDCG@5	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
CIKM	fine-tuning	0.8418	0.5202	0.4329	0.6163	0.4639	0.7257	0.4915	0.8891	0.5240
	PPR(light)	**0.8606**	**0.5399**	0.4445	**0.6427**	0.4777	**0.7562**	0.5064	**0.9110**	0.5372
	PPR(full)	0.8585	0.5396	**0.4476**	0.6409	**0.4802**	0.7522	**0.5084**	0.9068	**0.5392**
QQBrowser	fine-tuning	0.9593	0.7940	0.6459	0.8838	0.6751	0.9469	0.6912	0.9899	0.6999
	PPR(light)	0.9589	0.7866	0.6350	0.8812	0.6658	0.9471	0.6826	**0.9904**	0.6914
	PPR(full)	**0.9602**	**0.7965**	**0.6490**	**0.8863**	**0.6782**	**0.9486**	**0.6941**	0.9902	**0.7026**
AliEC	fine-tuning	0.8818	0.5708	0.4480	0.6912	0.4869	0.8057	0.5158	0.9317	0.5411
	PPR(light)	**0.8904**	**0.5729**	0.4465	**0.7032**	0.4886	**0.8173**	0.5177	**0.9426**	0.5428
	PPR(full)	0.8817	0.5726	**0.4519**	0.6934	**0.4908**	0.8050	**0.5190**	0.9308	**0.5443**
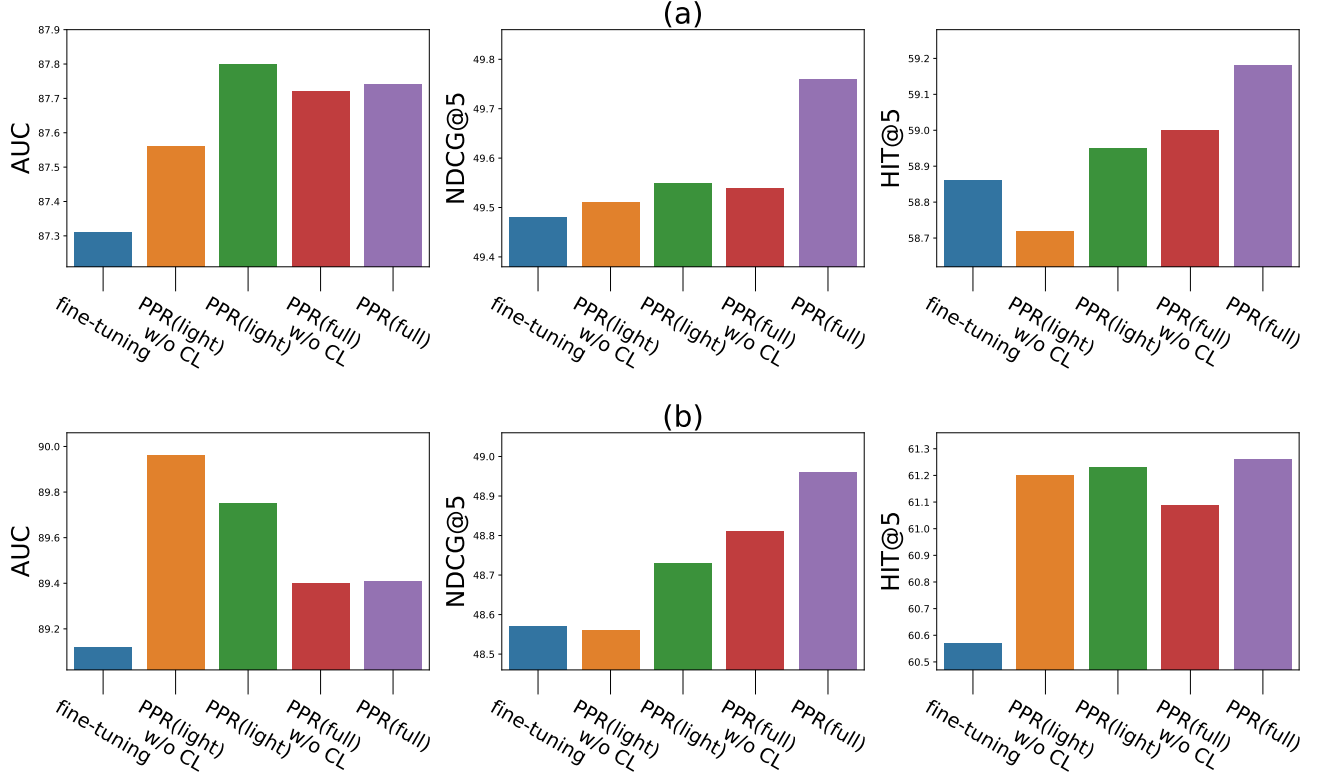
Fig. 3. Ablation study of PPR(light) and PPR(full) on (a) CIKM, and (b) AliEC. All improvements are significant over baselines (t-test with p < 0.05).

4.6 Ablation Study (RQ3)

In this section, we aim to confirm that the prompt-oriented contrastive learning is essential for PPR. Fig. 3 shows the results of different ablation versions of PPR(light) and PPR(full) on few-shot recommendation. We find that:

(1) Generally, the prompt-oriented CL brings consistent improvements on almost all metrics on both datasets except for AUC on PPR(light). It confirms the effectiveness of our prompt-oriented CL with two types of augmentations for better prompt-enhanced sequence modeling.

(2) PPR still outperforms fine-tuning on most metrics, even without the prompt-oriented CL. It verifies that our personalized prompt-tuning is truly superior to fine-tuning in transferring pre-trained knowledge to downstream tasks. Besides, we deploy PPR on a CL-based model CL4SRec as the pre-training model and still achieve consistent improvements (in Sec. 4.7).

4.7 Universality of PPR (RQ4)

PPR is an effective and universal tuning framework, which can be easily deployed on different pre-training models. In this section, we further adopt PPR with CL4SRec [4] used as the pre-training model. The results of PPR models on few-shot recommendation based on CL4SRec are given in Table 6, from which we can find that:

(1) PPR models still achieve the best performances on all metrics in three datasets, which proves the universality of PPR with different pre-trained recommendation models. It implies that our PPR could function as a general enhanced tuning mechanism of fine-tuning when adopting large-scale pre-trained recommendation models for downstream tasks.

(2) Generally, PPR(full) outperforms PPR(light) on most metrics. It indicates that tuning item embeddings (i.e., the labels to be predicted) is essential in cold-start recommendation, which differs from the prompt-tuning in NLP. Nev-

TABLE 6
Results of PPR on few-shot recommendation based on CL4SRec. Improvements are significant (t-test with p<0.05).

| Dataset	Model	AUC	HIT@5	NDCG@5	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
CIKM	pre-train	0.8590	0.5865	0.4942	0.6699	0.5252	0.7612	0.5481	0.8960	0.5750
	fine-tuning	0.8660	0.5948	0.5035	0.6797	0.5309	0.7719	0.5542	0.9034	0.5804
	PPR(light)	0.8723	0.6013	0.5053	0.6891	0.5338	0.7851	0.5580	0.9106	0.5830
	PPR(full)	**0.8769**	**0.6066**	**0.5093**	**0.6982**	**0.5390**	**0.7928**	**0.5628**	**0.9157**	**0.5873**
QQBrowser	pre-train	0.9554	0.7817	0.6304	0.8779	0.6618	0.9426	0.6783	0.9863	0.6872
	fine-tuning	0.9623	0.8083	0.6593	0.8971	0.6883	0.9548	0.7030	0.9896	0.7101
	PPR(light)	0.9641	0.8077	0.6566	0.8989	0.6863	**0.9583**	0.7015	**0.9919**	0.7084
	PPR(full)	**0.9642**	**0.8138**	**0.6652**	**0.9008**	**0.6936**	0.9580	**0.7082**	0.9911	**0.7150**
AliEC	pre-train	0.8787	0.5878	0.4696	0.6977	0.5052	0.8021	0.5316	0.9248	0.5562
	fine-tuning	0.8912	0.6057	0.4857	0.7226	0.5234	0.8242	0.5491	0.9365	0.5716
	PPR(light)	**0.8992**	0.6114	0.4857	**0.7328**	0.5250	**0.8386**	0.5518	**0.9441**	0.5729
	PPR(full)	0.8934	**0.6140**	**0.4918**	0.7277	**0.5284**	0.8290	**0.5542**	0.9362	**0.5757**
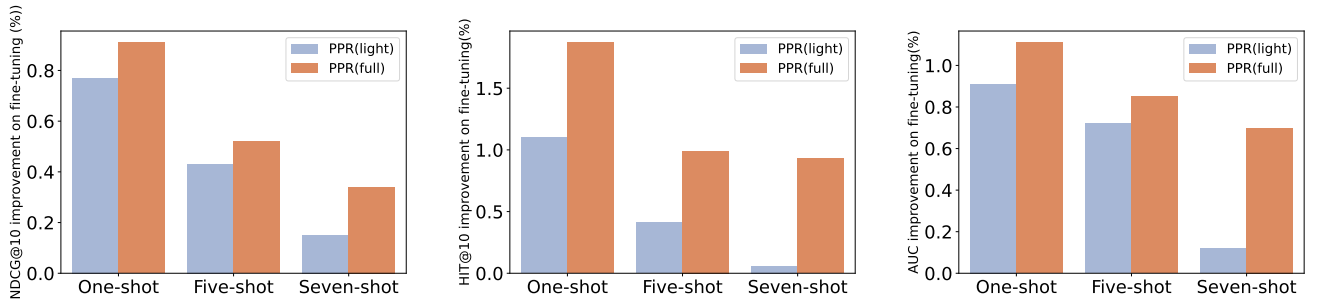
Fig. 4. Relative improvements of PPR models over fine-tuning with different degrees of data sparsity in CIKM dataset. All improvements are significant over baselines (t-test with p < 0.05).

ertheless, PPR(light) still performs better than fine-tuning on most metrics. Similar to Table 3, PPR(light) also has several better results in HIT@N. We can flexibly choose different implementations of PPR to enhance fine-tuning considering the practical demand on parameter-efficiency.

4.8 Model Analyses on Sparsity (RQ5)

We further explore the influence of data sparsity on PPR to show its robustness. Specifically, we crop all user behavior sequences of both train and test sets in tuning to represent different degrees of sparsity. We define the k-shot setting where all user behavior sequences are cropped and no longer than k, with $k = 1, 5, 7$. Fig. 4 displays the relative improvements of PPR(light) and PPR(full) over fine-tuning on AUC, HIT, and NDCG in CIKM. We have:

(1) Both PPR(light) and PPR(full) consistently outperform fine-tuning with different sparsity, which verifies the robustness of PPR in different cold-start scenarios.

(2) The improvements of PPR models increase with the maximum user behavior sequence length decreasing. It indicates that PPR can perform better on more sparse scenarios compared to fine-tuning. It is intuitive since the personalized prompts are more dominating when the behavioral information is sparser. In this case, the advantages of PPR (e.g., the better usage of pre-trained sequential modeling on prompt-enhanced sequence) are amplified.

4.9 Explorations of PPR on Other Tasks (RQ6)

Currently, pre-trained language models with prompt-tuning is dominating and widely verified as a paradigm in almost all NLP downstream tasks. Inspired by this, we believe that pre-trained recommendation model may also be a promising future direction for different recommendation tasks. Therefore, besides the cold-start recommendation task, we further explore other promising usages of personalized prompts on cross-domain recommendation and user profile prediction as an appropriate alternate of fine-tuning. Note that we **do not** want to rashly verify that PPR could achieve SOTA results on all recommendation tasks compared to well-designed sophisticated models. Instead, our explorations on other downstream tasks aim to shed light on the promising future direction of pre-training and prompt-tuning in recommendation.

4.9.1 PPR on Cross-domain Recommendation

Cross-domain recommendation (CDR) aims to transfer useful knowledge from the source domain to help the target domain [42], [43]. We focus on the CDR scenario with overlapping users and non-overlapping items. In this work, the source domain (pre-train set) is AliEC and the target domain (tuning set) is AliAD. For CDR, we directly use the user embeddings trained on the source domain by the same pre-trained model as our personalized prompts. Similarly, these source-domain user embeddings are also applied as side information for fine-tuning and PPR. We also implement (a)

TABLE 7
Results of cross-domain recommendation on AliEC→AliAD. All improvements are significant (t-test with p<0.01).

Dataset	Model	AUC	HIT@5	NDCG@5	HIT@10	NDCG@10	HIT@20	NDCG@20	HIT@50	NDCG@50
AliEC	SASRec(target)	0.6423	0.2752	0.2106	0.3590	0.2375	0.4594	0.2628	0.6568	0.3017
→	MiNet	0.6474	0.2573	0.1872	0.3499	0.2170	0.4507	0.2424	0.6561	0.2827
AliAD	fine-tuning	0.7165	0.3362	0.2494	0.4357	0.2813	0.5552	0.3115	0.7323	0.3465
	PPR(full)	**0.7343**	**0.3632**	**0.2799**	**0.4595**	**0.3109**	**0.5761**	**0.3402**	**0.7631**	**0.3772**

SASRec(target), which is only trained on the target domain without source information, and (b) MiNet [43], which is a classical CDR method, for comparisons. Note that there are no overlapping items, which means we have to tune the new target-domain item embeddings. Hence, we choose the PPR(full) version. The results are shown in Table 7.

We can find that PPR significantly outperforms all baselines on all metrics. The only difference between PPR and fine-tuning is the personalized prompt (all models use the same source/target domain features). It implies the promising application of PPR on other pre-trained tasks where items of pre-training and tuning stages are even different.

4.9.2 PPR on User Profile Prediction

User profile prediction task aims to predict users' profiles via their behavior sequences [7]. We conduct an exploration of PPR on this task by predicting users' graduate state in AliEC, which is a binary classification task. Precisely in PPR, the personalized prompts are generated by users' all profiles except the one to be predicted. For classification, we add a profile classifier after the final user representation for both PPR and fine-tuning. The results are shown in Table 8.

We can observe that PPR significantly outperforms fine-tuning on ACC, precision, and F1. It is because that PPR can better interact users' profile information with their behavioral information via a unified pre-trained sequential model. We also evaluate PPR on other user profile prediction tasks, such as gender and age, where PPR(light) has comparable results with far less parameter tuning. In the future, we can explore designing more customized prompts for different downstream tasks to improve their performances.

TABLE 8
Results of user profile prediction on AliEC. All improvements are significant (t-test with p<0.05).

Dataset	Model	ACC	Precision	Recall	F1
AliEC	fine-tuning	0.89	0.33	**0.82**	0.47
	PPR(light)	**0.92**	**0.42**	0.74	**0.54**

5 CONCLUSION AND FUTURE WORK

In this work, we propose a new personalized prompt-based recommendation to improve the tuning of pre-trained recommendation models. We conduct extensive experiments to verify the effectiveness, robustness, and universality of our PPR on cold-start recommendation tasks, and also explore the potential wide usage of PPR on other downstream tasks. PPR is the first small step on adopting personalized prompts instead of fine-tuning in pre-trained recommendation models, shedding lights on future explorations.

In the future, we will verify our PPR on more recommendation tasks with customized prompt-tuning manners, prompt generators and pre-training tasks. We will also explore the effectiveness of PPR on industrial extremely large-scale recommendation datasets and pre-trained models. Jointly considering pre-trained recommendation and language models also deserves further exploration.
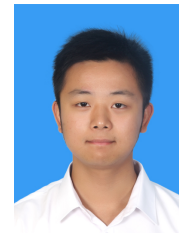
REFERENCES

[1] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proceedings of ICDM*, 2018.
[2] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of CIKM*, 2019.
[3] Z. Xie, C. Liu, Y. Zhang, H. Lu, D. Wang, and Y. Ding, "Adversarial and contrastive variational autoencoder for sequential recommendation," in *Proceedings of WWW*, 2021.
[4] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, and B. Cui, "Contrastive learning for sequential recommendation," in *ICDE*. IEEE, 2022.
[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL*, 2019.
[6] Z. Zeng, C. Xiao, Y. Yao, R. Xie, Z. Liu, F. Lin, L. Lin, and M. Sun, "Knowledge transfer via pre-training for recommendation: A review and prospect," *Frontiers in big Data*, 2021.
[7] C. Xiao, R. Xie, Y. Yao, Z. Liu, M. Sun, X. Zhang, and L. Lin, "Uprec: User-aware pre-training for recommender systems," *arXiv preprint*, 2021.
[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *NeurIPS*, 2020.
[9] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of EMNLP*, 2021.
[10] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification," *AI Open*, 2022.
[11] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *Proceedings of ACL*, 2021.
[12] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *ACL-IJNLP*, 2021.
[13] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," in *Proceedings of NAACL*, 2021.
[14] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.
[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of NIPS*, 2017.
[16] Y. Zheng, S. Liu, Z. Li, and S. Wu, "Cold-start sequential recommendation via meta learner," in *Proceedings of the AAAI*, 2021.

[17] Z. Liu, Z. Fan, Y. Wang, and P. S. Yu, "Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer," *Proceedings of SIGIR*, 2021.

[18] L. Chen, F. Yuan, J. Yang, X. He, C. Li, and M. Yang, "User-specific adaptive fine-tuning for cross-domain recommendations," *TKDE*, 2021.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint*, 2019.

[20] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *CVPR*, 2021.

[21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[22] K. Shin, H. Kwak, K.-M. Kim, M. Kim, Y.-J. Park, J. Jeong, and S. Jung, "One4all user representation for recommender systems in e-commerce," *arXiv preprint*, 2021.

[23] T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang *et al.*, "Self-supervised learning for large-scale item recommendations," in *Proceedings of CIKM*, 2021.

[24] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of SIGIR*, 2021.

[25] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, "S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization," in *CIKM*, 2020.

[26] F. Yuan, X. He, A. Karatzoglou, and L. Zhang, "Parameter-efficient transfer from sequential behaviors for user modeling and recommendation," in *Proceedings of SIGIR*, 2020.

[27] Y. Hou, S. Mu, W. X. Zhao, Y. Li, B. Ding, and J.-R. Wen, "Towards universal sequence representation learning for recommender systems," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 585–593.

[28] L. Li, Y. Zhang, and L. Chen, "Personalized prompt learning for explainable recommendation," *arXiv preprint arXiv:2202.07371*, 2022.

[29] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)," *RecSys*, 2022.

[30] Z. Cui, J. Ma, C. Zhou, J. Zhou, and H. Yang, "M6-rec: Generative pretrained language models are open-ended recommender systems," *arXiv preprint arXiv:2205.08084*, 2022.

[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," 2019.

[32] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint*, 2021.

[33] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," in *Proceedings of ACL*, 2022.

[34] T. Schick and H. Schütze, "Exploiting cloze questions for few shot text classification and natural language inference," *Proceedings of ACL*, 2020.

[35] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *TACL*, 2020.

[36] Y. Wu, R. Xie, Y. Zhu, F. Zhuang, A. Xiang, X. Zhang, L. Lin, and Q. He, "Selective fairness in recommendation via prompts," in *Proceedings of SIGIR*, 2022.

[37] Y. Gu, X. Han, Z. Liu, and M. Huang, "Ppt: Pre-trained prompt tuning for few-shot learning," *ACL*, 2022.

[38] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *Proceedings of the EMNLP*, 2020.

[39] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *UAI*, 2009.

[40] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *Proceedings of KDD*, 2019.

[41] J. Ding, G. Yu, X. He, Y. Quan, Y. Li, T.-S. Chua, D. Jin, and J. Yu, "Improving implicit recommender systems with view data." in *IJCAI*, 2018, pp. 3343–3349.

[42] G. Hu, Y. Zhang, and Q. Yang, "Conet: Collaborative cross networks for cross-domain recommendation," in *Proceedings of CIKM*, 2018.

[43] W. Ouyang, X. Zhang, L. Zhao, J. Luo, Y. Zhang, H. Zou, Z. Liu, and Y. Du, "Minet: Mixed interest network for cross-domain click-through rate prediction," in *Proceedings of CIKM*, 2020.

Yiqing Wu is currently pursuing his M.S. degree in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He recieved his B.E. degree from Dalian University of Technology in 2020. His main research interests include recommender system, graph neurnal network and data mining.


Ruobing Xie is a senior researcher of WeChat, Tencent. He received his BEng degree in 2014 and his master degree in 2017 from the Department of Computer Science and Technology, Tsinghua University. His research interests include recommender system, knowledge graph, and natural language processing. He has published over 60 papers in top-tier conferences and journals including KDD, WWW, SIGIR, ACL, AAAI, NeurIPS and TKDE. He is a member of CIPS Youth Working Committee and Social Media Processing Committee.

Yongchun Zhu is currently pursuing his Ph.D. degree in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His main research interests include transfer learning, fake news detection and recommender system. He has published over 20 papers in journals and conference proceedings including KDD, WWW, SIGIR, TNNLS and so on.

Fuzhen Zhuang is a professor in Institute of Artificial Intelligence, Beihang University. His research interests include transfer learning, machine learning, data mining, multi-task learning and recommendation systems. He has published over 100 papers in the prestigious refereed journals and conference proceedings, such as Nature Communications, TKDE, Proc. of IEEE, TNNLS, TIST, KDD, WWW, SIGIR, NeurIPS, AAAI, and ICDE.
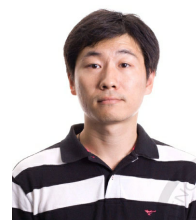
Xu Zhang received the master's degree in College of Computer Science and Engineering, Northeastern University in 2008. He is the team leader of WeChat User Profile, Tencent. His research interests include machine learning and its applications, such as social network, user interest model and user behavior research. He has published more than 10 papers in top conferences such as KDD, SIGIR, CIKM.

Leyu Lin received the masters degree in Institute of Computing Technology, Chinese Academy of Sciences, in 2008. He is currently the director of WeChat Recommendation Product Center, Tencent. His research interests include information retrieval and its applications, such as search system, recommendation system and computational advertising. He has published more than 20 papers in top conferences such as KDD, SIGIR, WWW.

Qing He received the BS degree from Hebei Normal University, Shijiazhuang, China, in 1985, the MS degree from Zhengzhou University, Zhengzhou, China, in 1987, both in mathematics, and the PhD degree in fuzzy mathematics and artificial intelligence from Beijing Normal University, Beijing, China, in 2000. He is a professor as well as a doctoral tutor with the Institute of Computing Technology, Chinese Academy of Science (CAS), Beijing, China, and he is a professor with the University of Chinese Academy of Sciences (UCAS), Beijing, China. His interests include data mining, machine learning, classification, and fuzzy clustering.