Meta Policy Learning for Cold-Start Conversational Recommendation

Zhendong Chu
University of Virginia
Charlottesville, VA, USA
zc9uy@virginia.edu

Hongning Wang
University of Virginia
Charlottesville, VA, USA
hw5x@virginia.edu

Yun Xiao
JD.COM Silicon Valley R&D Center
Mountain View, CA, USA
xiaoyun1@jd.com

Bo Long
JD.COM
Beijing, China
bo.long@jd.com

Lingfei Wu
JD.COM Silicon Valley R&D Center
Mountain View, CA, USA
lwu@email.wm.edu

ABSTRACT

Conversational recommender systems (CRS) explicitly solicit users' preferences for improved recommendations on the fly. Most existing CRS solutions count on a single policy trained by reinforcement learning for a population of users. However, for users new to the system, such a global policy becomes ineffective to satisfy them, i.e., the cold-start challenge. In this paper, we study CRS policy learning for cold-start users via meta reinforcement learning. We propose to learn a meta policy and adapt it to new users with only a few trials of conversational recommendations. To facilitate fast policy adaptation, we design three synergetic components. Firstly, we design a meta-exploration policy dedicated to identifying user preferences via a few exploratory conversations, which accelerates personalized policy adaptation from the meta policy. Secondly, we adapt the item recommendation module for each user to maximize the recommendation quality based on the collected conversation states during conversations. Thirdly, we propose a Transformer-based state encoder as the backbone to connect the previous two components. It provides comprehensive state representations by modeling complicated relations between positive and negative feedback during the conversation. Extensive experiments on three datasets demonstrate the advantage of our solution in serving new users, compared with a rich set of state-of-the-art CRS solutions.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Markov decision processes.*

KEYWORDS

Reinforcement Learning; Conversational Recommendation; Meta Learning

1 INTRODUCTION

While traditional recommendation solutions infer a user's preferences only based on her historically interacted items [4–6, 18, 33, 34, 43], conversational recommender systems (CRS) leverage interactive conversations to adaptively profile a user's preference [8, 26, 36]. The conversations in CRS focus on questions about users' preferences on item attributes (e.g., brands or price range), in the form of pre-defined question templates [11, 26, 36] or timely synthesized natural language questions [29, 51]. Through a series of question answering, a profile about a user's intended item can be depicted, even when the user is new to the system [8], i.e., the cold-start users, which gives CRS an edge in providing improved recommendations.

Christakopoulou et al. [8] first proposed the idea of CRS. Their solution focused on deciding what item to ask for feedback; and off-the-shelf metrics, such as upper confidence bound [1], were leveraged for the purpose. Following this line, reinforcement learning (RL) based methods become the mainstream solution recently for CRS. Sun and Zhang [36] built a policy network to decide whether to recommend an item, or otherwise which item attribute to ask about in each turn of a conversation. However, in these two early studies, the conversation is terminated once a recommendation is made, no matter whether the user accepts it or not. Lei et al. [26] studied multi-round conversational recommendation, where CRS can ask a question or recommend an item multiple times before the user accepts the recommendation (considered as a successful conversation) or quits (considered as a failed conversation). This is also the setting of our work in this paper. To better address multi-round CRS, Lei et al. [27] leveraged knowledge graphs to select more relevant attributes to ask across turns. Xu et al. [45] extend [26] by revising user embeddings dynamically based on users' feedback on attributes and items. And Deng et al. [11] unified the question selection module and the recommendation module in an RL-based CRS solution, which simplifies the training of CRS. However, all aforementioned RL-based methods rely on existing user embeddings to conduct conversations and recommendations, which are not applicable to new users.

Although CRS is expected to address the cold-start problem in recommendation, by profiling a new user via eliciting her preference about item attributes, how to acquire the most effective feedback to
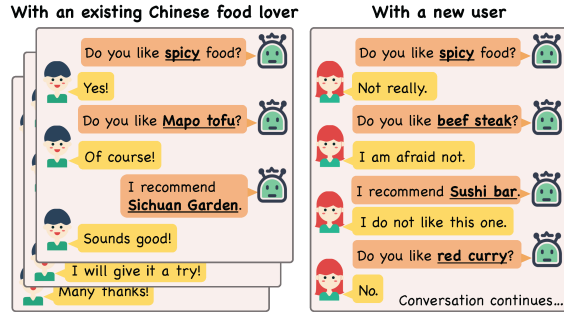
Figure 1: Example of cold-start CRS.

profile a single user still encounters the cold-start problem. More specifically, due to the heterogeneity of different users' preferences, the same policy can hardly be optimal in finding the sequence of interactions (asking questions or making recommendations) for all users, especially for those who do not contribute to the policy training. Consider the example shown in Fig.1, a policy trained with a population of Chinese food lovers cannot effectively serve new users who do not have any preferences on Chinese food. Once the interaction trajectory deviates from those often encountered during training, the effectiveness of the globally learnt CRS policy deteriorates, so does the quality of its recommendations.

We attribute this new challenge as cold-start policy learning in CRS, which is completely non-trivial but unfortunately ignored in most previous CRS studies. The goal is clear, i.e., adapt a CRS policy for each new user; but there are at least three main technical barriers blocking us from the goal. Firstly, *how to efficiently adapt a policy to new users?* The tolerance of users about a prolonged conversation or bad recommendations is limited [9, 10, 15, 28, 35], since all users wish to get high-quality recommendations with the least effort (e.g., shorter conversations) [37]. Hence, one cannot expect a large number of observations for CRS policy learning in a single user. Secondly, *how to effectively explore user preferences for policy adaptation?* As shown in Fig.1, successfully adapting a CRS policy to a new user depends on the user's preference, which however is elicited by the policy itself. This forms a *chicken-and-egg* dilemma [31] and adds another layer of consideration when acquiring user feedback: before identifying what item the user is looking for, one first needs to figure out what policy best suits for the inquiry. Thirdly, *how to decouple the adaptation of the conversation component and recommendation component in a CRS policy?* The conversation component (i.e., conversational policy) in CRS is to profile a user by actively eliciting her feedback, while the recommendation component (i.e., item recommender) is to identify the most relevant recommendations based on the profile. Adaptation in both components is needed for new users, but the strategy for adapting them could be different for respective goals.

In this paper, we address the problem of CRS policy learning for cold-start users via meta reinforcement learning [21, 31, 41], and name the proposed solution **MetaCRS**. For the first challenge, we propose to learn a meta policy for CRS from a population of users and adapt it to new users with only a few trials of conversational recommendations. The meta policy can be viewed as a starting point close to every single user's personalized policy. It thus builds the basis for efficient policy adaptation with only a handful of observations in each new user. Secondly, to acquire the most informative

feedback for policy adaptation, we design a meta-exploration policy to identify user preferences via a few exploratory conversations. Thirdly, in addition to the CRS policy, we also adapt the recommendation module in each user to maximize the recommendation quality. To support such a decoupled adaptation strategy, we design a Transformer-based [39] state encoder as the backbone, which communicates the training signals between the conversation and recommendation components.

To evaluate the effectiveness of the proposed model, we compared MetaCRS with several state-of-the-art baselines for CRS on three public datasets. The results strongly demonstrated the advantage of our solution in making satisfactory recommendations to new users in CRS with a reduced number of conversations. We also conducted extensive ablation analysis on each proposed component to inspect its contribution on the improved performance: 1) the meta-exploration policy elicit informative user feedback for fast policy adaptation; and 2) the adapted recommendation component makes better recommendations by cooperating with the adapted conversation component.

2 RELATED WORKS

Exploration-Exploitation (EE) Trade-off in CRS. CRS take advantage of conversations with users to elicit their preferences in real time for improved recommendations. The main research effort in CRS focuses on addressing the explore-exploit trade-off in collecting user feedback. The first attempt made by Christakopoulou et al. [8] employed multi-armed bandit models to acquire users' feedback on individual items. A follow-up study [47] set an additional bandit model to select attributes to collect user feedback and employed a manually crafted function to decide when to ask questions or make recommendations. Li et al. [30] unified attributes and items in the same arm space and let a bandit algorithm determine when to do what. Follow-up works [44, 49] also explored clustered and knowledge-aware conversational bandits.

Meta learning for recommendation. Meta learning [7, 13] has been widely used to solve the cold-start problem in recommender systems. Vartak et al. [38] studied the item cold-start problem (i.e., how to recommend new items to users). They proposed two adaptation approaches. One learns a linear classifier whose weights are determined by the items represented to the user before and adapts the classifiers' weights for each user. Another one learns user-specific item representations and adapts the bias terms in a neural network recommender for the purpose. Lee et al. [25] separated the representation layer and decision-making layer in a neural recommendation model, and executed local adaptation on the decision-making layer for each new user. Zou et al. [52] focused on interactive item recommendation, where the meta model is optimized by maximizing the cumulative rewards in each user. Kim et al. [23] deployed meta learning to online update recommender, where the meta learning rates are adaptively tuned on a per parameter and instance basis. To the best of our knowledge, we are the first to propose to tackle with cold-start CRS policy learning using meta reinforcement learning.

3 PRELIMINARY

In this section, we first formulate the problem of multi-round CRS as a reinforcement learning problem, and then illustrate the concept

of meta reinforcement learning and how we use it to address the cold-start challenge in CRS.

3.1 Problem Definition

In this work, we study the problem of multi-round conversational recommendation [26], where CRS can ask questions or make recommendations multiple times before the user accepts the recommendation or quits the conversation. Similar to traditional recommender systems, CRS face a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$; and we denote a specific user as u and a specific item as v. Each item v is associated with a set of pre-defined attributes $\mathcal{P}_v$. Attributes describe basic properties of items, such as movie genres in movie recommendations and authors in book recommendations.

We formulate the CRS problem by a Markov decision process (MDP) [11, 20, 27, 46], which can be fully described by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. $\mathcal{S}$ denotes the state space, which summarizes the conversation between the system and user so far. $\mathcal{A}$ denotes the action space for the system, which includes recommending a particular item or asking a specific attribute for feedback. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [-R_{max}, R_{max}]$ is a bounded reward function suggesting a user's feedback on the system's actions. As we focus on meta policy learning for CRS in this work, how to best define reward is not our objective. We follow the reward function defined in [11, 26, 27]. In particular, we include the following rewards: (1) $r_{\text{rec\_suc}}$, a large positive reward when the recommended item is accepted; (2) $r_{\text{rec\_fail}}$, a negative reward when the recommended item is rejected; (3) $r_{\text{ask\_suc}}$, a positive reward when the inquired attribute is confirmed by the user; (4) $r_{\text{ask\_fail}}$, a negative reward when the inquired attribute is dismissed by the user; (5) r_{quit}, a large negative reward when the user quits the conversation without a successful recommendation.

With this formulation, a conversation in CRS can be represented as $d = \{(a_1, r_1), ...(a_T, r_T)\}$, where T is the maximum number of allowed turns. A conversation (or an episode in the language of RL, which we will use exchangeably) will terminate when (1) the user accepts the recommended item; or (2) the agent runs out of maximum allowed turns. At each time step t, the CRS agent, which can be fully described by a policy $\pi(a_t|s_t)$, selects an action a_t based on the current state s_t. The training objective of a CRS policy is to maximize the expected cumulative rewards over the set of observed episodes D, i.e.,

$$\mathcal{L}(\pi) = - \mathbb{E}_{d \sim P(D)} \Big[\sum_{t=0}^{T} R_t \Big],$$

where $R_t = \sum_{t'=t}^{T} \gamma^{T-t'} r(a_t)$ is the accumulated reward from turn t to the final turn T, and $\gamma \in [0, 1]$ is a discount factor to emphasize rewards collected in a near term.

3.2 Meta Reinforcement Learning for CRS

Instead of learning a single global policy π, we propose to learn personalized policy π_u for each user u (new or existing) to address the cold-start challenge for CRS. The fundamental reason that almost all previous works [11, 26, 27] focused on global policy learning is that they (implicitly) assumed users know all attributes of their desired items and share the same responses over those attributes; in other words, user feedback is fully determined by the item. This assumption is unrealistically strong and naive, since different users

can describe the same item very differently, because of their distinct knowledge and preferences. For example, some users choose a mobile phone for its appearance while others choose it because of its brand. As a result, a global policy can hardly be optimal for every single user, especially the new users whose preferences are not observed during global policy training. In this work, we impose a weaker and more realistic assumption about users' decision making by allowing user-specific feedback $\mathcal{R}_{\mathcal{U}}$, which calls for personalized policies. Therefore, a personalized policy for user u should minimize,

$$\mathcal{L}_u(\pi) = - \mathbb{E}_{d \sim P(\mathcal{D}_u)} \Big[\sum_{t=0}^{T} R_u(a_t) \Big], \quad (1)$$

where $\mathcal{D}_u$ is a collection of conversations from user u and $R_u(a_t) = \sum_{t'=t}^{T} \gamma^{T-t'} r_u(a_{t'})$. To find the best personalized policy π_u (parameterized by θ_u) for each new user u, instead of learning from scratch every time, we choose to learn a meta policy parameterized by θ and use it as a starting point to look for θ_u. Following the convention of meta learning, we assume a set of conversations $\mathcal{D}_u^s$ (i.e., the support set) for policy adaptation , in addition to the set $\mathcal{D}_u^q$ (i.e., the query set) for policy evaluation. Hence, the size of support set $\mathcal{D}_u^s$ in each user u denotes the conversation budget for us to find θ_u when serving a single user. Given limited tolerance of an ordinary user to prolonged conversations, a performing solution should find the optimal θ_u with the size of $\mathcal{D}_u^s$ as small as possible. In the meta-train phase, we conduct local adaptation from the meta policy on the $\mathcal{D}_u^s$ of the existing users (i.e., training users), and then evaluate and update the meta policy on training users' $\mathcal{D}_u^q$. In the meta-test phase, we test the meta policy on the new users (i.e., testing users) by executing local adaptation on their support sets, and then test the obtained local policy on their query sets.
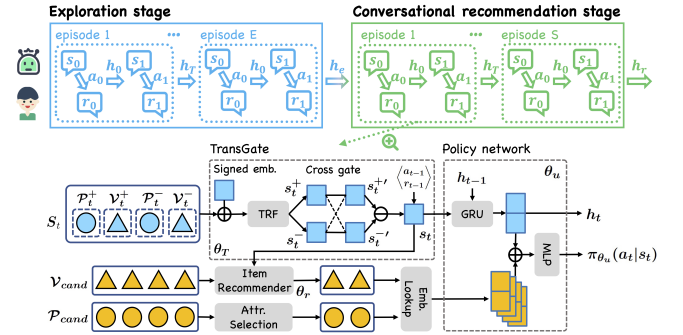

Figure 2: The workflow of MetaCRS training. Each user's support set is separated into the exploration stage and conversational recommendation stage. The last hidden state from the previous episode is passed to the next episode as its initial state throughout the course of MetaCRS in each user.

4 METHODOLOGY

In this section, we describe the design of MetaCRS in detail. We first introduce our two-stage meta policy learning framework designed for cold-start CRS. Then, we describe the details of the state-based item recommender, which is separately adapted to maximize the

recommendation quality. The Transformer-based state encoder, which aims to rapidly capture a user's preference from her both positive and negative feedback in a conversation, is lastly explained. Fig.2 shows the overview of MetaCRS.

4.1 Two-stage Meta Policy Learning for CRS

Motivated by the seminal work Model-Agnostic Meta-Learning (MAML) [13], we propose to first learn a meta CRS policy from a population of users; and then for each individual user, we adapt the meta policy to a user-specific policy with only a few trails of conversational recommendations with the user. We obtain the meta policy by maximizing the policy adaptation performance in a given set of training users. Specifically, in each user u, we perform policy adaptation on her support set $\mathcal{D}_u^s$, where θ_u is initialized with θ and then updated by optimizing Eq.(1) via gradient descent. The gradient for Eq.(1) is computed by the REINFORCE [42] algorithm,

$$\nabla_{\theta_u} \mathcal{L}_u(\pi_{\theta_u}) = -\mathbb{E}\Big[\sum_{t=0}^{T} R_u(a_t) \nabla_{\theta_u} \log \pi_{\theta_u}(a_t|s_t) \Big]. \quad (2)$$

The gradient of the meta policy with respect to θ (i.e., $\nabla_\theta \mathcal{L}_u(\theta_u)$) is computed in the same way as Eq.(2), but on the corresponding query set $\mathcal{D}_u^q$. In this way, the meta policy is optimized for generalization, akin to cross-validation. Note that to exactly compute the gradient for θ, we need to take a higher-order derivative in $\nabla_{\theta_u} \mathcal{L}_u(\theta_u)$ with respect to θ on the support set as well, since θ_u is a function of θ. In this work, we followed the the first-order approximation methods proposed in [13, 32] to simplify the gradient computation.

In meta-learning for supervised learning tasks, e.g., image classification [13, 32, 40], the support set and query set are predefined and thus not affected by the learnt models. Therefore, gradient-based optimization alone is sufficient for meta model learning and adaptation. But in our problem, what we will observe in $\mathcal{D}_u^s$ and $\mathcal{D}_u^q$ are completely determined by the employed policy π_{θ_u}, which however is supposed to be derived from $\mathcal{D}_u^s$ and $\mathcal{D}_u^q$. This causes the so-called *chicken-and-egg* dilemma [31] for meta policy learning which we discussed in the introduction, and calls for additional treatments beyond gradient-based policy optimization.

Potential bias in the currently learnt policy prevents it from being effective in acquiring the most informative feedback for meta policy learning and adaptation. Hence, we propose to separate policy adaptation in each user into an *exploration* stage and a *conversational recommendation* stage, and design their corresponding policies. To avoid ambiguity, we refer to the policy for the exploration stage as the meta-exploration policy (denoted as π_{θ_e}), and the policy for the conversational recommendation stage as the CRS policy (denoted as π_{θ_u}). This is similar to the explore-then-commit strategy [14, 24] in bandit literature. But note that our meta-exploration policy is *not* personalized, as its sole goal is to quickly identify what kind of user the system is interacting with. Hence, we choose to estimate it from the whole set of training users. In particular, we reserve the first few episodes in each user's support set for our exploration stage, denoted as the exploration set $\mathcal{D}_u^e$. The size of $\mathcal{D}_u^e$ is a hyper-parameter to be tuned for different CRS applications. $\mathcal{D}_u^e$ will only be used to estimate the meta-exploration policy π_{θ_e}.

In MetaCRS, the meta-exploration policy π_{θ_e}, meta CRS policy π_θ and personalized CRS policies $\{\pi_{\theta_u}\}_{u\in\mathcal{U}}$ are realized by the same RNN-based policy network architecture [12, 19, 41] with

Gated Recurrent Units (GRUs) [2, 12] to better encode the conversation history, but we estimate different parameters for them respectively. To avoid ambiguity, we will use the learning of meta-exploration policy π_{θ_e} as an illustrating example; and the same procedure applies to the learning of other policies.

Specifically, at the t-th turn of an episode, we observe a new state and encode it using a state encoder. We leave the discussion about our state encoding in Section 4.3. The encoded state s_t is provided as input to the policy network. The output h_t of the GRU is fed to a fully connected layer followed by a softmax function to produce the action distribution for $\pi_{\theta_e}(a_t|s_t)$. In each user, by the end of each episode, the GRU's last output hidden state h_T [1] is passed to the user's next episode as its initial state, such that this user's conversation history with the system can be continuously used to jump start her next conversation. Enabled by this design, information collected from the exploration stage is passed over to the conversational recommendation stage to profile who the user is (denoted as h_e), and then to the query set (denoted as h_r) to suggest what the user's preference could be. This sequential process is depicted in Fig.2.

The policy networks for π_θ and $\{\pi_{\theta_u}\}_{u\in\mathcal{U}}$ are trained via the meta learning procedure described at the beginning of this section, on top of the rewards defined in Section 3.1. But the meta-exploration policy π_{θ_e} is trained with a specially designed reward function, as its sole purpose is to identify what kind of user the system is serving. Inspired by [21, 22, 31], we adopt pre-trained user embeddings $\{e_u\}_{u\in\mathcal{U}}$ obtained on users with historical observations (i.e., training users) to design the exploration reward,

$$r_e(s_t) = \log P(e_u|s_t) - \log P(e_u|s_{t-1}), \quad (3)$$

where $P(e_u|s_t) = \frac{\exp(h_t^\top e_u)}{\sum_{u\in\mathcal{U}} \exp(h_t^\top e_u)}$. Note here we use the GRU's output hidden state h_t to predict the user embedding, just as how we use it to construct the policies. Specifically, we obtain $\{e_u\}_{u\in\mathcal{U}}$ from a Factorization Machine model [26] trained on observed user-item interactions in training users. More details about this user embedding learning can be found in Section 5.1. The insight behind our exploration reward design is that we promote the actions that help us identify a specific user during the exploration stage. Following the suggestion from [21, 22, 31], we also add a cross entropy loss on the meta-exploration policy network's latent state h_t to regularize the estimation of θ_e,

$$\mathcal{L}_e(\pi_{\theta_e}) = -\mathbb{E}\Big[\sum_{t=0}^{T} R_e(s_t) + \sum_{t=0}^{T} \log P(e_u|h_t) \Big], \quad (4)$$

where $R_e(s_t)$ is the accumulated discounted reward based on Eq.(3) from turn t. The gradient of the first term is also computed by the REINFORCE algorithm.

4.2 State-aware Item Recommender

Previous studies use a pre-trained recommender through the course of CRS [26, 27, 45], as their focus is mostly on deciding when to make a recommendation or otherwise what question to ask, i.e., the conversation component. A pre-trained recommender restricts the CRS policy to accommodate the recommender's behavior, which adds unnecessary complexity for policy adaptation. Such a black

---

[1]If the conversation ends before the maximum turn, h_T stands for the latent state at the successful recommendation.

box design slows down personalized policy learning. For example, a user wants a phone of a specific brand, but the recommender regards *brand* as an unimportant attribute. It is difficult for CRS to recommend successfully even though the policy already elicits her preferences, which will in turn hurts the policy adaptation since the episode is failed. Hence, it is crucial to also local adapt the recommender to learn to make high-quality recommendations in cooperation with the conversation component.

In MetaCRS, we set a learnable item recommender to rank candidate items based on the state embedding from the state encoder, which will be explained in Section 4.3. The ranking score of an item v is calculated by,

$$w_t(v) = e_v^\top (W_1 s_t + b_1),$$

where $\{W_1, b_1\}$ are learnable parameters for the recommender, collectively denoted as θ_r; s_t is the state embedding obtained from the state encoder. We perform local adaptation on θ_r to obtain a personalized recommender, by minimizing the following cross-entropy loss once a successful conversation concludes,

$$\mathcal{L}_r(\theta_r) = -\frac{1}{T_s}\mathbb{I}(T_s \le T)\sum_{t=0}^{T_s}\log\frac{\exp(w_t(v_s))}{\sum_{|\mathcal{V}_t^+|}\exp(w_t(v))}, \quad (5)$$

where T_s is the index of the successful turn and v_s is the accepted item. This loss function encourages the adapted recommendation component to identify the finally accepted item as early as possible in a conversation. We denote the meta parameters of θ_r as θ_R.

4.3 TransGate State Encoder

Previous solutions [45, 48, 50] have shown the power of negative feedback in CRS state modeling. It is even more important for cold-start CRS, especially in the early stage of policy adaptation when the policy is more likely to collect negative feedback. Ineffective modeling of negative feedback will slow down policy adaption. Moreover, positive and negative feedback posits distinct information about users' preference, and thus calls for different treatments. We employ a Transformer to model such complicated relations in an ongoing conversation into a state, with a cross gate mechanism to differentiate the impact from positive and negative feedback. We name this state encoder as TransGate.

At turn t, we accumulate four kinds of feedback from a user in this conversation: (1) $\mathcal{P}_t^+$, attributes confirmed by the user; (2) $\mathcal{V}_t^+$, candidate items satisfying all accepted attributes; (3) $\mathcal{P}_t^-$, attributes dismissed by the user; (4) $\mathcal{V}_t^-$, items rejected by the user. Collectively, we denote $\mathcal{S}_t = \{\mathcal{P}_t^+, \mathcal{V}_t^+, \mathcal{P}_t^-, \mathcal{V}_t^-\}$. We first map elements in $\mathcal{S}_t$ into vectors e with an embedding layer, where attribute and item embeddings are pre-trained with training users' historical observations. Candidate items and rejected items are aggregated separately to reduce the sequence length,

$$e_{\mathcal{V}}^+ = \frac{1}{|\mathcal{V}_t^+|}\sum_{v \in \mathcal{V}_t^+} e_v^+, \ e_{\mathcal{V}}^- = \frac{1}{|\mathcal{V}_t^-|}\sum_{v \in \mathcal{V}_t^-} e_v^-.$$

In the original Transformer [39], elements are encoded with position embeddings. In our case, the order among the elements is not important, but encoding the sign of user feedback (i.e., accepted or rejected) is critical. Inspired by position embeddings, we propose to encode user feedback into signed embeddings $\{e^+, e^-\}$. We add e^+ to positive elements and e^- to negative elements in $\mathcal{S}_t$. We use the

current candidate items to provide positive context. Then, we feed the obtained embeddings into L Transformer layers. For simplicity, we keep the notations of transformed embeddings unchanged. We then aggregate the positive and negative elements separately to obtain an embedding for positive feedback and an embedding for negative feedback,

$$s_t^+ = \frac{1}{1 + |\mathcal{P}_t^+|}\left(e_{\mathcal{V}}^+ + \sum_{p \in \mathcal{P}_t^+} e_p^+\right), \ s_t^- = \frac{1}{1 + |\mathcal{P}_t^-|}\left(e_{\mathcal{V}}^- + \sum_{p \in \mathcal{P}_t^-} e_p^-\right).$$

The positive and negative feedback embeddings may contain overlapped information, which will confuse policy learning. For example, an item that already satisfies all confirmed attributes so far can still be rejected by the user. We propose a cross gate mechanism to further differentiate the positive and negative information $s_t^{+\prime} = s_t^+ \odot g^-$, $s_t^{-\prime} = s_t^- \odot g^+$, where $\odot$ denotes the element-wise product and $\{g^+, g^-\}$ are defined as

$$g^+ = \sigma(W_2 s_t^+ + b_2), \ g^- = \sigma(W_3 s_t^- + b_3),$$

where $\sigma(\cdot)$ is the sigmoid function and $\{W_2, W_3, b_2, b_3\}$ are learnable parameters. We obtain the final state by $s_t = s_t^{+\prime} - s_t^{-\prime}$. The set of parameters for the TransGate encoder is denoted as θ_T, which is learnt from the conversations with training users. We should note once learnt this encoder is shared globally by all users without personalization. The state embedding is then concatenated with the encoding of $\langle a_{t-1}, r_{t-1}\rangle$ as the input to the RNN-based policy network. In particular, the action embedding is directly read off based on the pre-trained attribute and item embeddings, and we set a linear layer to encode the reward.

Algorithm 1: Optimization algorithm of MetaCRS

Input: User population $\mathcal{U}$, learning rates α, β, meta parameters $\theta, \theta_e, \theta_R, \theta_T$;

while *not* Done **do**

    Sample a batch of users $\mathcal{U}_b \sim P(\mathcal{U})$;

    for each $u \in \mathcal{U}_b$ **do**

        Collect $\mathcal{D}_u^e$ and h_e by executing π_{θ_e} ;

        Initialize $\theta_u = \theta, \theta_r = \theta_R$;

        Collect $\mathcal{D}_u^s$ and h_r by executing π_{θ_u} with h_e;

        Evaluate $\nabla_{\theta_u}\mathcal{L}_u$ and $\nabla_{\theta_r}\mathcal{L}_r$ using $\mathcal{D}_u^s$;

        Compute adapted parameters with gradient descent:

        $\theta_u = \theta_u - \alpha\nabla_{\theta_u}\mathcal{L}_u, \theta_r = \theta_r - \alpha\nabla_{\theta_r}\mathcal{L}_r$;

        Collect $\mathcal{D}_u^q$ by executing π_{θ_u} with h_r;

    end

    Update $\theta, \theta_R, \theta_T$ using each $\mathcal{D}_u^q$ by minimizing $\mathcal{L}_u, \mathcal{L}_r$;

    Update θ_e, θ_T using each $\mathcal{D}_u^e$ by minimizing $\mathcal{L}_e$;

end

4.4 Optimization Algorithm

Now we are finally equipped to illustrate the complete learning solution for MetaCRS in Algorithm 1. In the inner for-loop, we perform policy adaption to obtain the personalized CRS policy (including item recommender). In the outer while-loop, we update all meta parameters. To simplify the gradient computation, we stop the gradients on the inherited initial hidden state h_T from the latest episode in back-propagation. In practice, we update the local

parameters once an episode is executed, as we find empirically it works better than updating once after the whole $\mathcal{D}_u^s$ is finished. When serving new users in the meta-test phase, we fix $\{\theta_e, \theta_T\}$ and only execute local adaptation (the inner for-loop part in Algorithm 1) with the corresponding parameters initialized by $\{\theta, \theta_R\}$.

In each turn, we use all the candidate items $\mathcal{V}_{cand}$ (i.e., $\mathcal{V}_t^+$) and attributes $\mathcal{P}_{cand}$ to construct the action space, where $\mathcal{P}_{cand}$ is the entire attribute set excluding $\mathcal{P}_t^+$ and $\mathcal{P}_t^-$. Deng et al. [11] reported that a very large action space always slowed down policy learning. To generate a reasonable action space, we follow the manually crafted rules from [11] to select K_A attributes from $\mathcal{P}_{cand}$ and select the top-K_I items provided by the state-based item recommender.

5 EXPERIMENTS

To fully demonstrate the effectiveness of MetaCRS in solving the cold-start CRS problem, we conduct extensive experiments and study the following four research questions (RQ):

- **RQ1**: Can MetaCRS achieve better performance than state-of-the-art CRS solutions when handling new users?
- **RQ2**: Does our meta reinforcement learning based adaptation strategy work better than other adaptation strategies?
- **RQ3**: How quickly can MetaCRS obtain a good personalized policy for each user?
- **RQ4**: How does each proposed component contribute to the final performance of MetaCRS?

Table 1: Summary statistics of datasets.

	LastFM	BookRec	MovieLens
#Users	1,801	1,891	3,000
#Items	7,432	4,343	5,974
#Attributes	33	35	35
#Interactions	72,040	75,640	120,000
Avg. $|\mathcal{P}_u|$	7	8	12
Avg. $|\mathcal{P}_v|$	4.07	8.15	5.02
Avg. $|\mathcal{P}_o|$	5.44	5.30	4.25

5.1 Datasets

We evaluate MetaCRS on three multi-round conversational recommendation benchmark datasets [11, 26, 27, 48] and summarize their statistics in Table 1.

- **LastFM** [3] is for music recommendation. Lei et al. [26] manually grouped the original attributes into 33 coarse-grained attributes.
- **BookRec** [17] is for book recommendation. We further processed it by selecting top 35 attributes according to their TF-IDF scores across items and filter out items with too few attributes.
- **MovieLens** [16] is for movie recommendation. We performed the same pre-processing as on the BookRec dataset.

We randomly split users for training, validation and testing with the ratio 8:1:1, such that the evaluation set only contains new users. On each benchmark dataset, we obtained user, item and attribute embeddings (denoted as $e_{\mathcal{U}}, e_{\mathcal{V}}, e_{\mathcal{P}}$) using a variant of Factorization Machine (FM) proposed in [26] on observed user-item interactions in the training set. Similar to [11, 26, 27], we developed a user-simulator to generate conversations based on the observed user-item interactions in the dataset. We describe the simulator in detail in Section 5.2.1. However, the number of observed interactions in

each user is not even, which may cause the learned meta policy biased toward users with more observed interactions. To better study the problem of personalized CRS policy learning, as part of our simulation, we generated 40 user-item interactions for each user by sampling items proportional to the score $e_u^\top e_v$ to augment the interaction data for our evaluation purpose. We should note such simulation design will not ease the necessity of personalized CRS policy learning, since e_u is never directly disclosed to the policy. We provide our code and generated data to facilitate follow-up research and ensure the producibility of our reported results [2].

5.2 Experiment Settings

5.2.1 User simulator. CRS needs to be trained and evaluated via interactions with users. Previous simulator designs are item-centric [11, 26, 27], enforcing all users to respond in the same way to all attributes of target item v (i.e., confirming every entry in $\mathcal{P}_v$). This setting is unrealistically restrictive and eliminates the necessity of personalized policies. To demonstrate the utility of personalized CRS policy learning, we design a *user-centric* simulator that supports user-specific feedback in each conversation.

In detail, we used the pre-trained user and attribute embeddings to generate each user's preferred attribute set $\{\mathcal{P}_u\}_{u \in \mathcal{U}}$, by selecting the top-ranked attributes for each user based on the score $e_u^\top e_p$. During the course of CRS, the simulated user will only confirm the overlapped attributes in $\mathcal{P}_o = \mathcal{P}_u \cap \mathcal{P}_v$, and dismiss all others. On the BookRec dataset, because the original entries in $\mathcal{P}_v$ is too generic to be informative, i.e., too many attributes appear in almost all items, we decided to also increase $\mathcal{P}_v$ on this dataset by adding top-ranked attributes for each item based on the score $e_v^\top e_p$. We report the mean value of $|\mathcal{P}_u|$, $|\mathcal{P}_v|$ and $|\mathcal{P}_o|$ resulted from our simulation on each dataset in Table 1.

5.2.2 Baselines. To fully evaluate the effectiveness of MetaCRS, we compared it with a set of representative baselines. We categorized the baselines into three groups for different comparison purposes. In the first group, we compared MetaCRS with a rich set of state-of-the-art CRS methods to answer RQ1:

- **Max Entropy (MaxE)** is a rule-based method suggested in [26]. In each turn, the attribute with maximum entropy is to be asked or top-ranked items are to be recommended based on the rule.
- **EAR** [26] is a three-stage solution consisting estimation, action and reflection steps. It updates the conversation and recommendation components using reinforcement learning.
- **SCPR** [27] reformulates the CRS problem as an interactive path reasoning problem on the user-item-attribute graph. Candidate attributes and items are selected according to their relations with collected user feedback on the graph.
- **UNICORN (UNI)** [11] integrates the conversation and recommendation components into a unified RL agent. Two heuristics for pre-selecting attributes and items in each turn are proposed to simplify its RL training.

Baselines in this group rely on pre-trained user embeddings to make recommendations or compute states, which are not available in new users. To apply them to new users, we used the average embedding of all training users as the embedding for new users. This group of baselines are learnt on training users and then evaluated on the testing users.

---

[2]https://github.com/zdchu/MetaCRS.git

Table 2: Comparison of CRS performance among models on three datasets. * stands for the best performance in each group.

		MaxE	EAR	SCPR	UNI	ConUCB	ConTS	FPAN	UR	F-FT	F-IA	UR-FT	UR-IA	MetaCRS
LastFM	SR@10	0.137	0.428	0.432	0.441*	0.237	0.270	0.508	0.641*	0.533	0.529	0.613	0.678*	**0.713**
	AT	9.71	8.62	8.70	8.52*	8.69	8.93	8.08	7.02*	8.01	8.13	7.19	6.85*	**6.18**
BookRec	SR@10	0.206	0.320	0.329	0.358*	0.181	0.243	0.397*	0.384	0.405	0.411	0.417	0.420*	**0.487**
	AT	9.64	9.01	9.11	9.00*	9.52	9.17	8.31*	8.55	8.36*	8.52	8.54	8.41	**8.06**
MovieLens	SR@10	0.262	0.552	0.545	0.596*	0.272	0.434	0.589	0.681*	0.603	0.612	0.677	0.704*	**0.745**
	AT	9.46	7.98	7.89*	8.01	8.36	8.08	7.81	7.00*	7.78	7.69	7.14	6.88*	**6.27**

In the second group, we compared MetaCRS with solutions which handle new users by updating user embeddings dynamically within a conversation, such that they can provide adaptive recommendations. We consider the following algorithms:

- **ConUCB** [47] introduces the concept of super arms (i.e., attributes) to traditional bandit algorithms. Items and attributes with the highest upper confidence bound are selected. The attributes are asked in a fixed frequency by a hand-crafted function.
- **ConTS** [30] overcomes ConUCB's limitation by replacing the hand-crafted function with a Thompson sampling procedure. The user embeddings of cold-start users are updated with users' feedback on the asked attributes and items.
- **FPAN** [45] extends the EAR model by utilizing a user-item-attribute graph to enhance offline representation learning. User embeddings are revised dynamically based on users' feedback on items and attributes in the conversation.

Our TransGate and state-based item recommender can also dynamically capture user preferences, and provide adaptive recommendations within a conversation. To further study their value in learning personalized CRS policies, we integrated them with UNICORN, and denoted this variant as **UR**, which is also included in the second group. All baselines in this group used the same pre-trained embeddings as MetaCRS. To improve the practical performance of ConUCB and ConTS, we adopted the heuristics in [11] to pre-select arms according to the similarity with accepted attributes.

To answer RQ2, we equip FPAN and UR with the ability to adapt policies on new users via the following two widely used strategies, which forms the third group of baselines:

- **Fine-tuning (FT)**: We first pre-train a global policy on all training users. During testing, we fine-tune the policy on the whole support set of all new users.
- **Independent adaptation (IA)**: We first pre-train a global policy on all training users. For each new user, we perform continual training on her support set to obtain a personalized policy.

We denoted the resulted variants as **F-FT**, **F-IA** and **UR-FT**, **UR-IA** respectively. As we found policy gradient was more effective and efficient than UNICORN's original Q-learning based algorithm in our experiments, we applied policy gradient for model update in all UNICORN-based baselines.

5.2.3 Evaluation metrics. We followed the widely-used metrics in previous works [11, 26, 27] to evaluate the CRS solutions. We evaluated the average ratio of successful episodes within T turns by success rate (SR@T). We also evaluated average turns in episodes (AT). A better policy is expected to recommend successfully with less turns. The length of failed conversations is counted as T.

5.2.4 Implementation details. We performed the training of meta policy on training users, and local adaptation on validation and testing users. We selected the best model according to its validation performance. The query sets of testing users are used to obtain the final performance for comparison. We set the rewards as: $r_{rec\_suc} = 1$, $r_{rec\_fail} = -0.1$, $r_{ask\_suc} = 0.1$, $r_{rec\_fail} = -0.1$, $r_{quit} = -0.3$. The action embedding size and the hidden size are set to be 64 and 100, while reward embedding size is set to 10. We set 1 Transformer layer in the TransGate encoder. In MetaCRS, we took 5 episodes in the exploration stage and 10 episodes in the conversational recommendation stage by default. We set K_I, K_A and K_{rec} to 10. We performed standard gradient decent in local adaptation with a learning rate of 0.01, and updated the meta parameters using the Adam optimizer with a learning rate of 0.005 and L_2 regularization coefficient 1e-6. The discount factor γ is set to 0.999. To make a fair comparison, we run 15 episodes in new user for adaptation in the second group of baselines. The size of query set is fixed to 10. The maximum turn T in each episode is set to 10. We sample 5 users in each epoch when training MetaCRS. For all baselines, we used implementations provided by the papers and modified them as described before to support cold-start evaluation.

5.3 Overall Performance

We report the comparison results across all methods in Table 2. We can clearly observe that MetaCRS outperformed all baselines with large margins. First of all, the results of the first group of baselines confirmed a single global policy cannot handle new users. By learning personalized policies, MetaCRS showed advantages in the final recommendation performance. In the second group of baselines, bandit-based algorithms select actions according to simple linear models, which are not capable to capture complicated relations between the algorithms' actions and user feedback, especially when positive rewards are discrete and sparse. Hence, such solutions performed much worse than other deep learning based methods. Both FPAN and UR can provide adaptive recommendations like MetaCRS, and thus they outperformed all baselines in the first group. Interestingly, we observe that FPAN outperformed EAR considerably. Different from EAR, FPAN updates user embeddings dynamically with users' positive and negative feedback on attributes and items by two gate modules, which enable dynamic item recommendation as in MetaCRS. This improved performance proves the necessity of adaptive recommendation during the course of CRS. UR showed general improvement against UNICORN and FPAN, both of which rely on a fixed FM model to recommend items (UNICORN also uses the FM model to pre-select items as actions) when performing policy training. Our state-based item recommender is able to provide

improved recommendations at each time step by utilizing training signal from accepted items, which brings concrete benefits.

In the third group, with the adaptation on new users, FT and IA led to general improvement, which indicates the necessity of policy adaptation in new users. Specifically, IA outperformed FT in most cases, which again proves that personalized polices are beneficial. Even though FPAN is able to dynamically capture users' preference with its gating modules, such knowledge did not generalize well on new users, which limited the improvement in F-FT and F-IA. By adapting both conversation and recommendation components in CRS, MetaCRS and UR-IA showed improvements against other baselines, which validates the necessity of our decoupled adaptation strategy. But as UR is not trained for generalization, it is hard to find optimal personalized policies for new users starting from such a global policy in UR-FT and UR-IA, while the meta model in MetaCRS is trained for generalization. In addition, the meta-exploration policy provides useful information for fast adaptation. Thus MetaCRS is able to perform better even with fewer adaptation episodes (first 5 episodes are used to explore user preferences).

5.4 Ablation Study

5.4.1 Impact of support set size. Since policy adaptation is performed on the support set, it is important to study how many episodes are needed to obtain a good personalized policy (RQ3). In this experiment, we gradually increased the size of support sets with a step size 5. We kept the size of exploration episodes unchanged since 5 episodes are empirically sufficient for pinning down the target user's preference. Due to space limit, we only reported the results on the LastFM and BookRec dataset in Figure 3, and similar results were also observed on the MovieLens dataset. With a larger support set, the success rate increases considerably and the number of average turn also reduces. This is expected since more observations can be collected to better adapt the meta policy for each user. And this result also demonstrates the promise of personalized CRS policy learning: the quality of recommendation increases rapidly as the users get engaged with the system, which leads to a win-win situation for both users and system.
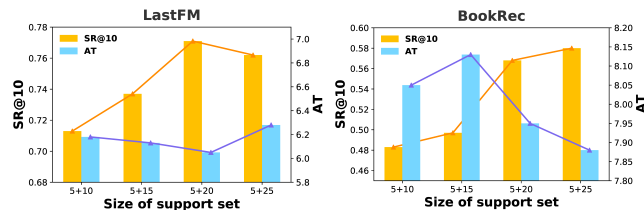

Figure 3: Performance comparisons w.r.t. size of support set.

5.4.2 Impact of different MetaCRS components. In this section, we study the contribution of different components in MetaCRS to answer RQ4. Firstly, we evaluated the model's performance without local adaptation, which essentially evaluated the learnt meta policy. Secondly, we removed the meta-exploration policy and directly executed policy adaptation. This setting shows how a dedicated exploration strategy affects policy adaptation. Finally, we replace the TransGate module with a linear layer similar to [50] to study how state representation learning affects the CRS performance. In

particular, the positive and negative embeddings are obtained by taking the average of all positive and negative feedback separately.

Table 3: Ablation analysis in MetaCRS.

	LastFM		BookRec		MoiveLens	
	SR@10	AT	SR@10	AT	SR@10	AT
¬adaptation	0.632	6.68	0.378	8.56	0.630	7.31
¬exploration	0.677	6.64	0.411	8.37	0.738	6.65
¬TransGate	0.678	6.41	0.428	8.51	0.724	6.95
MetaCRS	**0.713**	**6.18**	**0.487**	**8.06**	**0.745**	**6.27**

We present the results in Table 3. Firstly, we can observe the performance before adaptation is not bad, or even better than most of our baselines in Table 2, which suggests the meta policy in MetaCRS already captured some important patterns for interacting with users. We can further compare the learnt meta policy with UR in Table 2, which shares the same state encoder and item recommender, but was trained globally. UR is slightly better than the meta policy in MetaCRS. The reason is UR is trained to maximize performance on training users and generalized by the i.i.d. assumption. But the meta policy is trained to maximize the adapted policies' performance, not its own performance on new users. Hence, when testing users share reasonable similarity with training users, UR can be effective in serving the testing users. But we can observe a large performance gain after adaptation, which proves the meta policy successfully serves as a good starting point for fast adaptation. Next, it is clear that without the exploration stage the performance degenerates. It confirms recognizing who the system is serving is critical for a successful adaptation. We finally evaluate the effectiveness of TransGate, without which the performance degenerates on all three datasets. This demonstrates the necessity of fine-grained modeling of user feedback, especially the negative feedback, for understanding users' preferences.

6 CONCLUSION

In this work, we present a meta reinforcement learning based solution to handle the problem of CRS policy learning in cold-start users. We learn a meta policy for generalization and fast adapt it on new users. We developed three components to ensure the efficiency and effectiveness of policy adaptation. First, a dedicated meta-exploration policy is adopted to identify the most informative user feedback for policy adaptation. Second, an adaptive state-aware recommendation component is built to quickly improve recommendation quality for new users. Third, a Transformer-based state encoder implicitly models positive and negative feedback collected in a conversation to precisely profile user preference.

Currently our policy adaptation is performed independently across users; to further reduce its sample complexity, collaborative policy adaptation among users can be introduced to leverage observations among both new and existing users. As previous works reported improved CRS performance using a knowledge graph (KG), it is also interesting to study how personalized policy learning can benefit from KGs, e.g., adapts the entity relations in each user or design the meta-exploration strategy based on the KG.

7 ACKNOWLEDGEMENT

REFERENCES

[1] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

[4] Renqin Cai, Xueying Bai, Zhenrui Wang, Yuling Shi, Parikshit Sondhi, and Hongning Wang. 2018. Modeling sequential online interactive behaviors with temporal point process. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 873–882.

[5] Renqin Cai, Qinglei Wang, Chong Wang, and Xiaobing Liu. 2020. Learning to structure long-term dependence for sequential recommendation. *arXiv preprint arXiv:2001.11369* (2020).

[6] Renqin Cai, Jibang Wu, Aidan San, Chong Wang, and Hongning Wang. 2021. Category-aware collaborative sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 388–397.

[7] Dong Chen, Lingfei Wu, Siliang Tang, Xiao Yun, Bo Long, and Yueting Zhuang. 2022. Robust Meta-learning with Sampling Noise and Label Noise via Eigen-Reptile. *Proceedings of the 39th International Conference on Machine Learning* (2022).

[8] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD*. 815–824.

[9] Zhendong Chu, Jing Ma, and Hongning Wang. 2021. Learning from Crowds by Modeling Common Confusions.. In *AAAI*. 5832–5840.

[10] Zhendong Chu and Hongning Wang. 2021. Improve learning from crowds via generative augmentation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 167–175.

[11] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified Conversational Recommendation Policy Learning via Graph-based Reinforcement Learning. *arXiv preprint arXiv:2105.09710* (2021).

[12] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* (2016).

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. PMLR, 1126–1135.

[14] Aurélien Garivier, Tor Lattimore, and Emilie Kaufmann. 2016. On explore-then-commit strategies. *Advances in Neural Information Processing Systems* 29 (2016).

[15] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline a/b testing for recommender systems. In *Proceedings of the 11st ACM WSDM*. 198–206.

[16] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm TIIS* 5, 4 (2015), 1–19.

[17] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[19] John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79, 8 (1982), 2554–2558.

[20] Mengdi Huai, Jianhui Sun, Renqin Cai, Liuyi Yao, and Aidong Zhang. 2020. Malicious attacks against deep reinforcement learning interpretations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 472–482.

[21] Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. 2019. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424* (2019).

[22] Pierre-Alexandre Kamienny, Matteo Pirotta, Alessandro Lazaric, Thibault Lavril, Nicolas Usunier, and Ludovic Denoyer. 2020. Learning adaptive exploration strategies in dynamic environments through informed policy regularization. *arXiv preprint arXiv:2005.02934* (2020).

[23] Minseok Kim, Hwanjun Song, Yooju Shin, Dongmin Park, Kijung Shin, and Jae-Gil Lee. 2022. Meta-Learning for Online Update of Recommender Systems. (2022).

[24] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

[25] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. Melu: Meta-learned user preference estimator for cold-start recommendation. In *Proceedings of the 25th ACM SIGKDD*. 1073–1082.

[26] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th WSDM Conference*. 304–312.

[27] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive path reasoning on graph for conversational recommendation. In *Proceedings of the 26th ACM SIGKDD*. 2073–2083.

[28] Lihong Li, Jin Young Kim, and Imed Zitouni. 2015. Toward predicting the outcome of an A/B experiment for search relevance. In *Proceedings of the Eighth ACM WSDM Conference*. 37–46.

[29] Raymond Li, Samira Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Proceedings of the 32nd NeurIPS Conference*. 9748–9758.

[30] Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2021. Seamlessly unifying attributes and items: Conversational recommendation for cold-start users. *ACM TOIS* 39, 4 (2021), 1–29.

[31] Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. 2021. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *ICML*. PMLR, 6925–6935.

[32] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[33] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[34] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[35] Tobias Schnabel, Paul N Bennett, Susan T Dumais, and Thorsten Joachims. 2018. Short-term satisfaction and long-term coverage: Understanding how users tolerate algorithmic exploration. In *Proceedings of the 11st ACM WSDM*. 513–521.

[36] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st ACM SIGIR*. 235–244.

[37] Franck Tétard and Mikael Collan. 2009. Lazy user theory: A dynamic model to understand user selection of products and services. In *2009 42nd Hawaii International Conference on System Sciences*. IEEE, 1–9.

[38] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. (2017).

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[40] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. 2019. Multimodal model-agnostic meta-learning via task-aware modulation. *arXiv preprint arXiv:1910.13616* (2019).

[41] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* (2016).

[42] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[43] Jibang Wu, Renqin Cai, and Hongning Wang. 2020. Déjà vu: A contextualized temporal attention mechanism for sequential recommendation. In *Proceedings of The Web Conference 2020*. 2199–2209.

[44] Junda Wu, Canzhe Zhao, Tong Yu, Jingyang Li, and Shuai Li. 2021. Clustering of Conversational Bandits for User Preference Learning and Elicitation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2129–2139.

[45] Kerui Xu, Jingxuan Yang, Jun Xu, Sheng Gao, Jun Guo, and Ji-Rong Wen. 2021. Adapting User Preference to Online Feedback in Multi-round Conversational Recommendation. In *Proceedings of the 14th ACM WSDM*. 364–372.

[46] Fan Yao, Renqin Cai, and Hongning Wang. 2021. Reversible action design for combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:2102.07210* (2021).

[47] Xiaoying Zhang, Hong Xie, Hang Li, and John CS Lui. 2020. Conversational contextual bandit: Algorithm and application. In *Proceedings of The Web Conference 2020*. 662–672.

[48] Yiming Zhang, Lingfei Wu, Qi Shen, Yitong Pang, Zhihua Wei, Fangli Xu, Bo Long, and Jian Pei. 2021. Multi-Choice Questions based Multi-Interest Policy Learning for Conversational Recommendation. *arXiv preprint arXiv:2112.11775* (2021).

[49] Canzhe Zhao, Tong Yu, Zhihui Xie, and Shuai Li. 2022. Knowledge-aware Conversational Preference Elicitation with Bandit Feedback. In *Proceedings of the ACM Web Conference 2022*. 483–492.

[50] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1040–1048.

[51] Kun Zhou, Yuanhang Zhou, Wayne Xin Zhao, Xiaoke Wang, and Ji-Rong Wen. 2020. Towards Topic-Guided Conversational Recommender System. In *Proceedings of the 28th International Conference on Computational Linguistics*. 4128–4139.

[52] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural interactive collaborative filtering. In *Proceedings of the 43rd ACM SIGIR*. 749–758.