Rethinking Conversational Recommendations: Is Decision Tree All You Need?

A S M Ahsan-Ul Haque
ah3wj@virginia.edu
Department of Computer Science, University of Virginia
Charlottesville, VA, USA

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

ABSTRACT

Conversational recommender systems (CRS) dynamically obtain the users' preferences via multi-turn questions and answers. The existing CRS solutions are widely dominated by deep reinforcement learning algorithms. However, deep reinforcement learning methods are often criticized for lacking interpretability and requiring a large amount of training data to perform.

In this paper, we explore a simpler alternative and propose a decision tree based solution to CRS. The underlying challenge in CRS is that the same item can be described differently by different users. We show that decision trees are sufficient to characterize the *interactions* between users and items, and solve the key challenges in multi-turn CRS: namely *which questions to ask*, *how to rank the candidate items*, *when to recommend*, and *how to handle user's negative feedback on the recommendations*. Firstly, the training of decision trees enables us to find questions which effectively narrow down the search space. Secondly, by learning embeddings for each item and tree nodes, the candidate items can be ranked based on their similarity to the conversation context encoded by the tree nodes. Thirdly, the diversity of items associated with each tree node allows us to develop an early stopping strategy to decide when to make recommendations. Fourthly, when the user rejects a recommendation, we adaptively choose the next decision tree to improve subsequent questions and recommendations. Extensive experiments on three publicly available benchmark CRS datasets show that our approach provides significant improvement to the state of the art CRS methods.

CCS CONCEPTS

• **Information systems** → **Users and interactive retrieval**; **Recommender systems**.

KEYWORDS

Conversational recommender system; interactive retrieval

ACM Reference Format:
A S M Ahsan-Ul Haque and Hongning Wang. 2022. Rethinking Conversational Recommendations: Is Decision Tree All You Need?. In *Proceedings of the 31st ACM Int'l Conference on Information and Knowledge Management (CIKM '22), Oct. 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557433

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10.
https://doi.org/10.1145/3511808.3557433

1 INTRODUCTION

Whereas traditional approaches in recommender systems infer user preferences solely based on historical user-item interaction data [12, 30, 31], conversational recommender systems (CRS) elicit user preferences through interactive question answering [7, 20, 22, 39]. The traditional approaches are insufficient in adapting to the user's ongoing information need, since the user's preference can deviate over time from historical data. Moreover, users may look for different items under different situations. For example, a user who is looking for restaurant recommendations may seek a specific type of food at a particular time, or may consider other factors such as wait-time, availability of parking and etc differently when making her choices. A CRS solution (often referred to as "agent") can profile the user's current preference by the feedback collected from its strategically planned questions [17, 36].

Earlier forms of CRS can be traced back to interactive recommender systems [3, 11, 33] and critiquing-based recommender systems [24, 27, 28]. In the CRS setting proposed by Christakopoulou et al. [5], the agent searches for the target item using a multi-armed bandit algorithm solely in the item space, i.e., keep recommending different items. Zhang et al. [39] expanded the domain to the attribute space so that the agent needs to predict two things: what questions to ask and then which items to recommend, i.e., a series of questions followed by a recommendation. Li et al. [15] further expanded the setting by deciding between asking a question or making a recommendation at each round of interaction. As a result, the agent can ask a series of questions about item attributes and make several rounds of recommendations in an interleaved manner. This is often referred to as the multi-turn CRS setting and also the focus of our work.

Current multi-turn CRS solutions are dominated by deep reinforcement learning algorithms [8, 13, 14, 35]. Although encouraging empirical performance has been reported in literature, we question whether a simpler alternative exists and can achieve comparable or even better performance, as the deep models are often criticized for their lack of interpretability and high demand of training data to perform. Intuitively, decision tree is a natural choice to construct the questions in multi-turn CRS [26, 41], where each question narrows down the candidate set of items. However, naively using a decision tree to partition the item space for the purpose of CRS is infeasible, because different users can view the same item differently and thus provide distinct answers to the same question even when they are looking for the same item. Figure 1 shows this issue with an example in restaurant recommendation. From the observed user-item interactions, we notice that for the same burger shop, one user would describe it by its "low price" aspect, while another user did not pay attention to its price aspect but focuses on

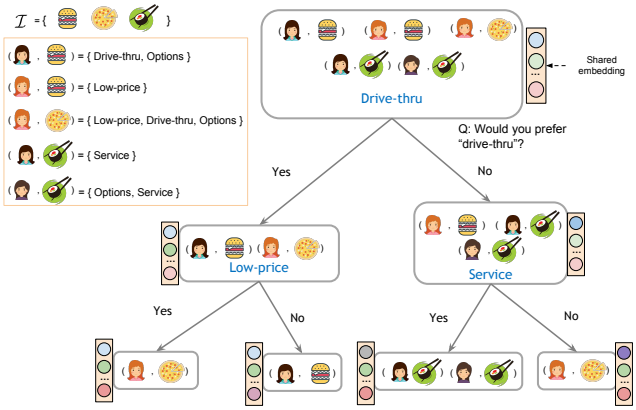

Figure 1: Overview of user-item interaction tree in FacT-CRS. We can learn shared embedding of user-item interaction using the interaction tree and use it to ask questions and recommend.

its "options" and "drive-through" service. As the partition of item space is exclusive in a decision tree, i.e., the items are separated into non-overlapping groups at the same level of tree nodes, if we used the question "price" to locate the restaurant in our interactions with the two users, we will surely fail to find the burger shop for at least one of them.

The problem described above motivates us to instead *partition the user-item interactions*: the matching between a user-item pair during the course of CRS can be characterized by how the user would describe the item. As a result, a path on the decision tree groups the common descriptions about items from different users, and different descriptions of the same item then result in different paths on the tree. As shown in Figure 1, the burger shop is now placed in different tree nodes to reflect the fact that different users would describe it differently. This sheds light on the possibility of using a decision tree for multi-turn CRS.

In this paper, we propose a rule-based solution to multi-turn CRS, namely FacT-CRS (Factorization Tree based Conversational Recommender System). We use user-item interactions to guide us to build a set of decision trees, which will be referred to as user-item interaction trees. Whereas the existing CRS methods [8, 13, 14, 35] rely on user embeddings to make recommendations, we propose to construct user-item interaction embeddings. There are still three main challenges remaining to address in order to complete the solution. Firstly, *how to rank the items?* Clearly a decision tree naturally forms the questions. However, as shown in Figure 1, each node in the decision-tree may contain a varying number of items. When the number of candidate items is more than what we can recommend in a turn, we need to decide the ranking of items before making a recommendation. We extend the concept of factorization tree [26] to learn embeddings for user-item interactions and all the candidate items while constructing the decision trees. For all observed user-item interaction pairs located in a tree node (i.e., the users provided the same answers to the questions asked so far about their intended items), an embedding vector is assigned to match against the item embeddings. During training, the embedding vectors are learnt in a way such that the matched items will be ranked higher for the associated user-item pairs than those unmatched ones. At inference time, when we decide to make a recommendation at a tree node, we will use the corresponding interaction embedding to rank all candidate items.

Secondly, *when to make a recommendation?* It is desirable that the agent makes a recommendation as soon as it is confident, before the user's patience wears out. It motives us to make a recommendation before exhausting the questions in the path of the interaction tree. An appropriate turn to make a recommendation is when 1) asking further questions does not provide much information gain, and 2) when the number of candidate items is small enough. Based on those considerations, we propose two strategies to make early recommendations. When building the interaction tree, we keep track of how much information gain is achieved using the Gini Index and we stop splitting a node if the information gain is below a threshold. At inference time, we can make recommendations before reaching the leaf node, when the number of unique items associated on the node is less than a threshold.

Thirdly, *how to handle a user's negative feedback about the recommended items?* Online feedback is an important aspect of multi-turn CRS where the agent should improve its action during a conversation, not only when the the user answers the planned question but also when he/she rejects a set of recommendations. When the agent encounters a rejection, it becomes apparent that 1) the rejected items are not what the user is looking for, and 2) the target item is still on the lower part of the ranked list. Based on this insight, we design an adaptive feedback module to refine the inferred interaction embeddings based on the rejected items before moving to the next round of interaction.

To evaluate the effectiveness of FacT-CRS, we conducted extensive experiments on three benchmark CRS datasets: LastFM, BookRec and MovieLens. The experiment results demonstrate that FacT-CRS performed significantly better in finding the target items for the users using fewer turns. Extensive analysis shows that FacT-CRS improved the conversation by narrowing down the candidate items taking user-item interaction into account and adapting to the online feedback effectively.

2 RELATED WORKS

CRS leverages multiple turns of questions and answers to obtain user preferences in a dynamic manner. There have been mainly four lines of research in CRS. In the item-based approaches, the agent keeps making recommendations based on users' immediate feedback. Christakopoulou et al. [5] proposed this line of research which marked the inception of CRS. They employed multi-armed bandit models to acquire the users' feedback on individual items, such that model updates itself at each turn.

Later in the question-driven setting, the domain was expanded so that the system needed to predict two things: 1) what questions about item attributes to ask, and 2) which items to recommend. Note that, in this setting the system only recommends by the end of conversation. Zhang et al. [39] proposed a model which consisted of three stages. In the initiation stage, user initiates the conversation. In the conversation stage, the system asks about the user preferences on attributes of items. And in the final stage, the system

recommends the items. Zou et al. [43] proposed Qrec which asks questions in a predetermined number of times and then makes recommendation once. In Qrec, they chose the most uncertain attribute to ask (the attribute that the system has the least confidence between positive and negative feedback). Christakopoulou et al. [4] later studied the setting where the user can provide multiple answers (for example: "choose all the attributes that you like."). There is another approach in CRS, which synthesize natural language. This adds personalized response to the conversation and it is applicable to an even broader scope. Usually, in this approach, there is a natural language understanding module and a natural language generation module. This approach is beyond the scope of our work.

In this paper, we focus on the multi-turn setting of CRS, where an agent needs to choose between asking a question or making a recommendation in each turn of conversation. Lei et al. [13] expanded the single-turn recommendation CRS to the multi-turn setting, where multiple questions and recommendations can be made in one conversation until the user accepts the recommendation or until the end of the conversation.

Reinforcement learning (RL) has been widely adopted in multi-turn CRS, which formulates the CRS problem as a Markov Decision Process (MDP) [19, 23, 40]. Recent works on RL-based interactive recommendation [29, 34, 37, 42, 44] have been shown to effectively recommend items by modeling the users' dynamic preferences. The objective of these methods is to learn an effective RL policy to decide which items to recommend. Sun et al. [25] used a belief tracker based on an LSTM model to determine when to recommend, but their model was not able to handle when user rejected a recommendation. Lei et al. [13] utilized three different modules: the estimation module predicts user preference on items and attributes; the action module decides whether to ask about attributes or to recommend items; and the reflection module updates the model when there is negative feedback. A dynamic preference model was introduced by Xu et al. [35], where they proposed a gating mechanism to include both positive and negative user feedback. Deng et al. [8] combined the question selection and recommender modules. They proposed two heuristics to reduce the candidate action space by pre-selecting attributes and items in each turn. Zhang et al. [38] used a bandit algorithm to select attributes and used a heuristic to decide whether to ask questions about attributes or make recommendations. Li et al. [16] unified attributes and items in the same space in a multi-armed bandit setting to determine the questions, and used another bandit model to determine when to recommend. Li et al. [15] used a deep RL based model to decide when to make a recommendation or which question to ask. Chu et al. [6] developed a meta reinforcement learning based solution to handle new users in CRS.

Although RL-based models dominate modern CRS solutions, in this paper, we explore a simpler and effective alternative based on decision trees. The RL methods provide strong baselines to compare the recommendation quality of our work.

3 METHODOLOGY

In this section, we first describe the problem of multi-turn conversational recommendation. Our work is motivated by finding a simple decision tree based model to solve the challenges in multi-turn conversational recommendation. We first explain how a decision tree can effectively model the user-item interactions to capture the potential matching between a user and an item, which translates to a set of questions to filter and rank items for recommendations. Then we describe how to adapt the user-item interaction tree in FacT-CRS to effectively address the challenges in CRS.

3.1 Preliminary

Multi-turn CRS. In this paper, we study the multi-turn conversational recommendation problem, since it has been popularly adopted because of its realistic setting [8, 13, 35]. In this setting, the agent takes multiple turns to ask questions regarding item attributes or make recommendations (e.g., movies, restaurants etc.). We denote the set of items as $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$, and the set of users as $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$. The set of attributes $\mathcal{F} = \{f_1, f_2, \ldots, f_p\}$ are used to describe the items. Each item is associated with a set of predefined attributes $\mathcal{F}_i$. Suppose that a user $u \in \mathcal{U}$ is in a conversation with the CRS agent and her target item is $i^+ \in \mathcal{I}$. Each conversation constitutes of multiple turns. At each turn t, the agent needs to decide whether to ask a question or to make a recommendation. Depending on the agent's action, each turn can either be 1) a question asked by the agent and followed by the user's answer, or 2) top-K recommendations followed by the user's acceptance or rejection. In the question-answer turn, the agent asks an attribute $f_t \in \mathcal{F}$, i.e., "do you prefer attribute f_t?" In the recommendation turn, the user accepts the recommendation, if the target item i^+ is contained in the top-K recommendations. Otherwise, the user rejects the recommendation. The conversation is considered successful, if the user accepts a recommendation. Otherwise, the conversation fails, if the agent reaches a maximum turn limit. The goal of CRS is to successfully recommend items to the user with the fewest number of turns.

User-item interaction. We use the pair (u, i) to denote that user u interacted with item i in history. We use $\mathcal{R}$ to denote the set of user-item interactions. Let the number of interaction $|\mathcal{R}| = q$. Each user-item interaction $\mathbf{r}_{u,i} \in \{0, 1\}^p$, where $(u, i) \in \mathcal{R}$. Here, 1 denotes that the attribute is mentioned in the interaction between the user and item (e.g., the user uses this attribute to describe the item), and 0 denotes that the attribute is not mentioned. An example could be: user u describes item i using the terms of $\{f_1, f_2\}$, as shown in Figure 1. Here, the interaction content $\mathbf{r}$ would be a p dimensional vector description with 1 for f_1, and f_2 and 0 for the remaining attributes.

3.2 User-item Interaction Tree

User-item interaction tree (or just "interaction tree" for short) is the building block of our FacT-CRS model that we use for asking questions, narrowing down the candidate set of items, and ranking the candidate items. The goal of the interaction tree is to allow us to hierarchically learn the interaction embeddings as a function of the attributes $\mathcal{F}$. We use FacT [26] to associate each interaction and each item with a d-dimension vector. The embeddings of all the interactions and items are respectively denoted as $\mathbf{S} \in \mathbb{R}^{q \times p}$ and $\mathbf{V} \in \mathbb{R}^{n \times p}$. Formally, we associate each interaction $(u, i) \in \mathcal{R}$ with an embedding $\mathbf{s}_{u,i} \in \mathbb{R}^d$ and each item i with an embedding $\mathbf{v}_i \in \mathbb{R}^d$. Intuitively, we want that a) the dot product $\mathbf{s}_{u,i}^T \mathbf{v}_i$ is maximized and

b) $\mathbf{s}_{u,i}^T \mathbf{v}_j$ is minimized when $j \neq i$. We achieve this using the cross-entropy loss defined in the following,

$$\mathcal{L}_{CE}(\mathbf{S}, \mathbf{V}, \mathcal{R}) = \sum_{(u,i)\in\mathcal{R}} \log\left(1 - \sigma\left(\mathbf{s}_{u,i}^T \mathbf{v}_i\right)\right) \tag{1}$$

Here, $\sigma(.)$ is the sigmoid function. To differentiate the relative ordering among candidate items, we also introduce the Bayesian Pairwise Ranking (BPR) objective [21]. The intuition is that the items that a given user has interacted with should be scored higher than the items the user did not interact with. We sample the negative set $\mathcal{D}_{u,i}^{neg}$ for interaction (u, i) by choosing i_{neg} where $(u, i_{neg}) \notin \mathcal{R}$. The BPR loss is calculated as in Eq. (2),

$$\mathcal{B}\left(\mathbf{s}_{u,i}, \mathbf{V}, \mathcal{D}_{u,i}^{neg}\right) = \sum_{i_{\text{neg}}\in\mathcal{D}_{u,i}^{\text{neg}}} \log\sigma\left(\mathbf{s}_{u,i}^T \mathbf{v}_i - \mathbf{s}_{u,i}^T \mathbf{v}_{i_{\text{neg}}}\right) \tag{2}$$

Each node z in the interaction tree is associated with a question $f_z \in \mathcal{F}$ (i.e., "Do you prefer attribute f_z?"). Let us consider a subset of all interactions $\mathcal{R}_z \subseteq \mathcal{R}$. Based on the user-item interaction (u, i) content $\mathbf{r}_{u,i}$, we can check whether or not a specific attribute f_l is mentioned. For each attribute f_l, we can partition the subset of user-item interactions $\mathcal{R}_z$ into two disjoint sets $\mathcal{R}_{z_{+|f_l}}$ and $\mathcal{R}_{z_{-|f_l}}$ as given in Eq. (3),

$$\begin{aligned} \mathcal{R}_{z_{+|f_l}} &= \left\{(u,i) \mid r_{u,i_l} = 1, \mathbf{r}_{u,i} \in \mathcal{R}_z\right\} \\ \mathcal{R}_{z_{-|f_l}} &= \left\{(u,i) \mid r_{u,i_l} = 0, \mathbf{r}_{u,i} \in \mathcal{R}_z\right\} \end{aligned} \tag{3}$$

All the user-item interactions in the same node share the same description path from the root node and thus will share the same interaction embedding. Let $\mathcal{R}_{z_{+|f_l}}$ and $\mathcal{R}_{z_{-|f_l}}$ be the observed user-item interactions in the resulting partitions, and $\mathbf{s}_{pos}, \mathbf{s}_{neg}$ be the corresponding user-item interaction embeddings in each of the partitions. Note that $\mathbf{s}_{pos}$ is the shared embedding of all the user-item interactions in $\mathcal{R}_{z_{+|f_l}}$ and $\mathbf{s}_{neg}$ is the shared embedding of all the user-item interactions in $\mathcal{R}_{z_{-|f_l}}$. We can find both embeddings by solving the following optimization problem:

$$\begin{aligned} \mathcal{L}(f_l|\mathcal{R}_z) = &\min_{\mathbf{s}_{pos},\mathbf{s}_{neg},\mathbf{V}} \mathcal{L}_{CE}(\mathbf{s}_{pos}, \mathbf{V}, \mathcal{R}_{z_{+|f_l}}) + \mathcal{L}_{CE}(\mathbf{s}_{neg}, \mathbf{V}, \mathcal{R}_{z_{-|f_l}}) \\ &+ \lambda_{BPR}\left(\sum_{(u,i)\in\mathcal{R}_{z_{+|f_l}}} \mathcal{B}\left(\mathbf{s}_{u,i}, \mathbf{V}, D_{u,i}^{neg}\right) + \sum_{(u,i)\in\mathcal{R}_{z_{-|f_l}}} \mathcal{B}\left(\mathbf{s}_{u,i}, \mathbf{V}, D_{u,i}^{neg}\right)\right) \\ &+ \lambda_s\left(\|\mathbf{s}_{pos}\|_2 + \|\mathbf{s}_{neg}\|_2\right) \end{aligned} \tag{4}$$

where λ_{BPR} is a coefficient that controls the trade-off between cross-entropy loss and BPR loss, $\|.\|$ is the L2 regularization to reduce model complexity, and λ_s is the regularization coefficient.

An optimal attribute split would partition the user-item interaction subset $\mathcal{R}_z$, where the embeddings in each disjoint group minimize the loss as given by Eq. (4). We can find the optimal attribute f_z by exhaustively searching through the attribute set $\mathcal{F}$ using Eq. (5):

$$f_z = \arg\min_{f_l\in\mathcal{F}} \mathcal{L}(f_l|\mathcal{R}_z) \tag{5}$$

We build the interaction tree by recursively splitting each partition until the maximum depth H_{max} is reached. Once constructed, the user-item interaction tree allows us to infer the interaction embedding for a new interaction at inference time by following the tree structure.

Candidate items. Each node in the user-item interaction tree maintains a subset of observed interactions. Since each interaction is a user-item pair, the items in the interaction set of that node form the candidate items for recommendation. Formally, given the subset of interaction $\mathcal{R}_z$ at node z, the candidate set of items is given by:

$$\mathcal{I}_z = \{i|(u,i) \in \mathcal{R}_z\} \tag{6}$$

Recommendation score. Suppose, after traversing the user-item interaction tree, the inferred interaction embedding is $\mathbf{s}_t$. The recommendation score of each candidate item at turn t can be readily computed as:

$$w_t^{\mathcal{I}}(i) = \mathbf{s}_t^T \mathbf{v}_i \tag{7}$$

3.3 FacT-CRS: User-item Interaction Tree for Conversational Recommendation

Successfully solving the multi-turn CRS problem requires addressing the following four problems: namely, 1) *which questions to ask*, 2) *how to rank the candidate items*, 3) *when to recommend*, and 4) *how to handle user's negative feedback on the recommendations*. Based on our construction of interaction tree described in Section 3.2, the first two questions are partially addressed. And in this section, we mainly focus on the remaining two key questions, based on the structure of a decision tree.

3.3.1 Which questions to ask and how to rank the candidate items? User-item interaction tree is learnt to optimize the questions to be asked based on user feedback, i.e., following the paths on the tree to narrow down the search space. However, given the maximum depth of a trained decision tree is fixed, we cannot ask more than H_{max} questions using one tree. For our model to generalize, we need to ask an arbitrary number of questions. Random forest [2] is an ensemble learning method that provides a feasible way to solve this challenge by creating multiple trees. We build a random forest of N user-item interaction trees. We can build the interaction trees in parallel so that each interaction tree in the forest considers a maximum of f_{max} attributes where $f_{max} \leq q$. These attributes are randomly sampled from the complete attribute set $\mathcal{F}$. We use τ_j to denote the j^{th} interaction tree and $\mathcal{T}$ to denote the set of all interaction trees, therefore an interaction forest. We will discuss how to leverage these multiple trees to realize multi-turn CRS in Section 3.3.3.

As each tree node is associated with an interaction embedding vector, once the agent decides to make a recommendation, it will use Eq (7) to compute the ranking scores of all candidate items and return the top-K items as the recommendation of this turn.

3.3.2 When to recommend? The interaction tree gives us a natural way to decide when to recommend, i.e., when reaching the leaf node of the tree. But this restricts us to ask at most H_{max} questions before we can start a recommendation turn. However, an important goal of conversational recommender systems is to minimize the number of interactions with the user. In this case, it is desirable to make a recommendation when the agent is "confident" about the item to be recommended. Hence, the question becomes: how to make

an early recommendation? We use the following two strategies to recommend early.

• During training (pruning): When building each user-item interaction tree, we stop splitting a node when the items in that node is "homogeneous", i.e., most of the interactions are about only a few items. We use Gini Index [9] for this purpose. Suppose, for an internal node z in an interaction tree, the set of items from the user-item interaction $\mathcal{R}_z$ is $\mathcal{I}_z$ as given by Eq. (6). We calculate Gini Index of node z as:

$$G_z = 1 - \left(\frac{|\mathcal{I}_z|}{|\mathcal{R}_z|}\right)^2 \tag{8}$$

If the Gini Index of a node is greater than a predetermined threshold γ, we stop further splitting that node.

• During testing (EarlyRec): At inference time, a small number of items in a node is a strong indication that it is time to recommend. If we encounter a node in the user-item interaction tree that has no more than η items, we can make an early recommendation. Let the set of items in that node be $\mathcal{I}_z$. Then, if $|\mathcal{I}_z| \leq \eta$, we make an early recommendation. If strictly $|\mathcal{I}_z| \leq K$, then we can include all the items in $\mathcal{I}_z$ in the recommendation. We rank all the items $\in \mathcal{I} \setminus \mathcal{I}_z$ using the scoring function in Eq. (7). The remaining $K - |\mathcal{I}_z|$ items with the highest scores are included in the top-K recommendation.

3.3.3 How to handle online negative feedback? Handling negative feedback or recommendation rejection is an important challenge in CRS. When the user rejects a recommendation, it provides us a strong signal to improve the subsequent questions and recommendations. To effectively handle negative feedback from a user's rejected recommendations, we approach this problem in three steps: retaining information from previously asked user-item interaction trees, effectively choosing the next user-item interaction tree, and updating the predicted user-item interaction embedding.

• Retaining information from the previously asked user-item interaction trees. We need to move to a new user-item interaction tree when the user rejects our recommendation.

Suppose, at turn t, we have asked questions from the set of interaction trees $\mathcal{T}_t^\alpha$, and we have obtained the set of user-item interaction embeddings $\mathcal{S}_t^\alpha$. We keep the conversation history by updating the inferred interaction embedding to be the mean of the visited user-item interaction embeddings $\mathcal{S}_t^\alpha$. As a result, the updated interaction embedding at turn t becomes:

$$\mathbf{s}_t = \frac{1}{|\mathcal{S}_t^\alpha|} \sum_{\mathbf{s}_j \in \mathcal{S}_t^\alpha} \mathbf{s}_j \tag{9}$$

• Choosing the next user-item interaction tree. Assume that we have finished traversing an interaction tree and made a recommendation. If the user rejects, how do we select the next tree? One way is to randomly pick an unvisited tree. However, we believe that we can do better by using the following strategy. Since the predicted interaction embedding is expected be close to the target interaction embedding, we propose to move to the closest tree first. Suppose, at turn t, we have asked questions from the set of $\mathcal{T}_t^\alpha$ user-item interaction trees, and have asked the set of attributes $\mathcal{F}_t^\alpha \subseteq \mathcal{F}$. Based on Eq. (9), let the current inferred user-item interaction embedding be $\mathbf{s}_t$. We traverse the remaining user-item interaction trees in parallel by using only the attributes in $\mathcal{F}_t^\alpha$, i.e, the attributes we already know the answers to. Assume that we get the interaction embedding s'_j from the remaining interaction tree $\tau_j \in \mathcal{T} \setminus \mathcal{T}_t^\alpha$. We score each of the remaining tree τ_j according to how similar the corresponding interaction embedding $\mathbf{s}'_j$ is to the current interaction embedding $\mathbf{s}'_t$, as given by Eq. (10),

$$w_t^{\mathcal{T}}(\tau_j) = \mathbf{s}_j'^T \mathbf{s} \tag{10}$$

Using this closest tree first strategy, we choose the user-item interaction tree τ_j with the highest similarity score as the next tree to continue the conversation.

• Updating the predicted user-item interaction embedding. We first identify the set of items that cause the failure and then update the inferred interaction embedding accordingly. Assume at turn t we make a recommendation using the interaction embedding $\mathbf{s}_t$, which the user rejected. We denote this set of recommended items as $\mathcal{I}_r$. We use the following two strategies to handle online feedback: 1) As $\mathcal{I}_r$ was rejected by the user, we update the predicted user-item interaction embedding to penalize the items in $\mathcal{I}_r$. 2) Since the target item is still expected to have high scores, we promote the items in the next top-K set. Based on $\mathbf{s}_t$, denote the next K highest scoring items as set $\mathcal{I}_p$.

Combining both strategies, we update the user-item interaction embedding after a rejected recommendation as follows:

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \frac{\alpha_p}{|\mathcal{I}_p|} \sum_{i \in \mathcal{I}_p} \mathbf{v}_i - \frac{\alpha_n}{|\mathcal{I}_r|} \sum_{i \in \mathcal{I}_r} \mathbf{v}_i \tag{11}$$

Here, α_r and α_p are hyper-parameters that respectively determine how much we penalize the items in $\mathcal{I}_r$ and how much we promote the next top-ranked items $\mathcal{I}_p$. Algorithm 1 presents the inference steps of a single conversation session.

4 EXPERIMENTS

To evaluate the effectiveness of FacT-CRS in solving the challenges in CRS, we perform quantitative experiments guided by the following research questions (RQ):

- **RQ1.** Can a rule-based method i.e., FacT-CRS, achieve better performance than the state-of-the-art RL-based methods in the multi-turn CRS setting?
- **RQ2.** How much improvement does early recommendation strategy offer compared to asking questions from the entire interaction tree?
- **RQ3.** Is online update of inferred interaction embedding useful in making better recommendations?

4.1 Dataset

We evaluate the recommendation quality of FacT-CRS on three benchmark datasets used in multi-turn CRS.

- **LastFM** [1] is a dataset for music artist recommendation. Lei et al. [13] pruned the users with fewer than 10 reviews [12, 21] to reduce data sparsity, and processed the original attributes by combining synonyms and removing low frequency attributes. They categorized the original attributes into 33 coarse-grained attributes.
- **BookRec** [18] This is a book recommendation dataset filtered by removing low frequency TF-IDF attributes and keeping the top 35 attributes.

Algorithm 1 Inference in FacT-CRS

Require: Interaction Forest $\mathcal{T}$, interaction content $\mathbf{r}_{u,i} \in \{0,1\}^p$;
Ensure: Outcome of conversation, either Success or Failure;

```
Current tree T_t selected via cross-validation;
Visited trees T_t^α = {};
Asked attributes F_0^α = {};
Inferred interaction embeddings S_0^α = {};
for turn t = 1, 2, ..., T do
    From T_t ask f_z, and get I_z, s_z;
    while f_z in F_t^α do
        From T_t ask f_z, and get I_z, s_z;
    end while
    F_t^α = F_{t-1}^α ∪ {f_z};
    if |I_z| ≥ η and z is not leaf node then
        continue;
    end if
    Get s_t from S_t = S_{t-1}^α ∪ {s_z} via Eq. (9);
    I_r = top-K items using s_t and I_z via Eq. (7);
    if User accepts I_r then
        return Success;
    else
        Update s_t via Eq. (11);
        for each remaining tree τ_j ∈ T \ T_t^α do
            answer questions in F_t^α and get s'_j;
        end for
        T_{t+1} = highest scoring tree via Eq. (10);
    end if
end for
return Failure
```

- **MovieLens** [10] is a movie recommendation dataset filtered by keeping top 35 attributes according to their TF-IDF scores.

The statistics of the datasets are shown in Table 1. We randomly split the users into disjoint groups of 8:1:1 for training, validation and testing users. The code and data used in the experiments are available at https://github.com/HCDM/XRec.

Table 1: Summary of datasets.

	LastFM	BookRec	MovieLens
# Users	1,801	1,891	3,000
# Items	7,432	4,343	5,974
# Interactions	72,040	75,640	120,000
# Attributes	33	35	35

4.2 Experiment Settings

4.2.1 User simulator. Conversational recommendation is a dynamic process for user preference elicitation. Similar to [8, 13, 14, 25, 35, 39], we created a user-simulator to enable the CRS training and testing. In each conversational session, an observed user-item pair (u, i) is first selected. We call the item i the target item or the ground-truth item for that conversation. Previous simulators [8, 13, 14, 35] assume that all of item i's attributes $\mathcal{F}_i$ is the oracle set of attributes preferred by the user in this session. This means that all users will respond in the same way to the selected item. This setting is, however, unrealistic, because in reality, every user may not equally value every attribute of an item. Hence, this design eliminates the potential of personalized responses.

We design a user-based simulator that can handle user-specific feedback in each conversation round in the following way. Each item i is associated with a set of attributes $\mathcal{F}_i$, and each user u has a preferred attribute set $\mathcal{F}_u$. $\mathcal{F}_i$ and $\mathcal{F}_u$ are uniformly sampled for each item i and user u before the simulation starts. For a user-item interaction pair (u, i), our simulator only accepts (responds "Yes" to) an attribute f_l if and only if it is mentioned in $\mathcal{F}_i \cap \mathcal{F}_u$; otherwise it will respond "No". We set the maximum turn limit in a conversation to 10. The user leaves the conversation after the turn limit is reached. We set K = 10, so that we are limited to recommend only 10 items in a recommendation turn.

4.2.2 Evaluation Metrics. Following [8, 13, 14, 25, 35, 39], we use success rate and average turn as evaluation metrics. We use the success rate at turn T (SR@T) to measure the ratio of successful conversations. In an interaction where user u interacted with item i^+, we call i^+ the ground-truth or the target item. A session of conversation is successful if the agent can identify the ground-truth item. We also report the average turns (AT) needed to end the round of conversation. The number of turns in a failed conversation is set to the the maximum turn limit T. The quality of recommendation is greater for larger SR@T, whereas the conversation is more efficient and to the point for smaller AT.

4.2.3 Implementation Details. We performed the training of FacT-CRS on the training users, and tuned the hyper-parameters of our model on the validation set of users. The best model was chosen based on the validation success rate. The testing users were used to obtain the final reported performance for comparison. We fixed the embedding dimension $d = 40$, $\lambda_{BPR} = 10^{-3}$, $\alpha_p = 10^{-3}$, $\alpha_n = 10^{-}2$, and $G_z = 0.996$. The depth of the interaction tree was chosen as 7.

4.2.4 Baselines. We evaluated the performance of FacT-CRS and compared with the following state-of-the-art multi-turn CRS baselines [8, 13, 14, 32, 35].

- **Max Entropy (MaxE)** [32]: In this method, the CRS agent chooses either an attribute to ask or top ranked items to recommend in a probabilistic way. The agent asks the attribute with the maximum entropy based on the past conversation history.
- **EAR** [13]: This is a three stage approach for multi-turn CRS: the estimation stage builds a predictive model to estimate user preference based on both items and attributes, the action stage learns a policy to decide whether to ask about attributes or make a recommendation, and reflection stage updates the recommendation model based on online user feedback.
- **FPAN** [35]: This extends EAR by dynamically revising user embeddings based on users' feedback. The relationship between attribute-level and item-level feedback signals are used to identify the specific items and attributes that causes the rejection of an item.
- **SCPR** [14]: It models CRS as an interactive path reasoning problem on a knowledge graph. It leverages target user's preferred attributes by following user feedback to traverse attribute graph. Using the knowledge graph enables it to reduce the search space of candidate attributes.

Table 2: Performance Comparison of different CRS models on three datasets. * represents the best performance among the baselines. The improvement over baseline is calculated against the best baseline values.

		MaxE	EAR	FPAN	SCPR	UNI	FacT-CRS	Improvement over baseline*
LastFM	SR@10	0.137	0.428	0.508*	0.432	0.441	0.719	41.53%
	AT	9.71	8.62	8.08*	8.70	8.52	6.65	17.69%
BookRec	SR@10	0.206	0.320	0.397*	0.329	0.358	0.438	10.33%
	AT	9.64	9.01	8.31*	9.11	9.00	8.23	0.96%
MovieLens	SR@10	0.262	0.552	0.589	0.545	0.596*	0.692	16.10%
	AT	9.46	7.98	7.81*	7.89	8.01	6.57	15.87%

- **UNICORN** [8]: This method integrates the conversation and recommendation components into a unified RL solution. UNICORN develops a dynamic weighted graph based RL method to learn a policy for selecting the action at each conversation turn. It pre-selects attributes and items to simplify the RL training.

All these methods rely on reinforcement learning models and pretrained user embeddings. To adopt them to new users at testing time, we use use the mean embedding of the train users.

4.3 Overall Performance

To answer RQ1, we first evaluate the recommendation quality of FacT-CRS in terms of success rate (SR@T) and average turn AT. A good CRS agent should be able to realize items which are more relevant to a user's preference and rank them higher in a result list.

We report the results of our experiments in Table 2. FacT-CRS consistently outperformed all baselines by a good margin on all datasets. FPAN performed better than the others among the baselines. We note that although it uses the same general structure as EAR, FPAN is able to generalize better on the new user because it updates the user embeddings dynamically with the user provided positive and negative feedback on attributes and items by two gated modules, which enables adaptive item recommendation. Although EAR is effective in handling large action space in CRS, its performance is limited by the separation of its conversation and recommendation components. UNICORN performs relatively better in this case through its usage of action selection strategy.

However, since all the baselines rely on user-level embedding, they cannot perform well on new test users. By hierarchically clustering the user-item interactions and exploiting the related user-item interactions, the interaction embeddings estimated by FacT-CRS more effectively capture the current preference of the user, even in new users. This enables its good empirical performance in our evaluation.

We also investigated how FacT-CRS compares to other baselines at each turn of the conversations. To investigate this, we compared the recommendation probability and success rate between FacT-CRS and FPAN on the LastFM dataset. As shown in Figure 2, FPAN can start recommending earlier, but when the user rejects a recommendation, it tries to recommend more items, instead of identifying the cause of failure by asking more questions.

In comparison, FacT-CRS handles the negative feedback more effectively. It asks more clarifying questions before making another recommendation. FacT-CRS first tries to identify the negative items and then uses Eq. (11) to update the predicted user-item interaction embedding, and then moves on to the next tree. This allows

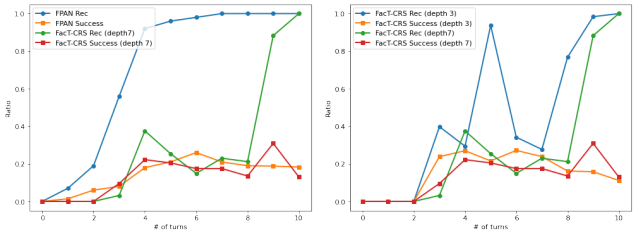

Figure 2: (Left) Ratio of recommendation and recommendation success rate at each turn on LastFM dataset (FacT-CRS vs FPAN). (Right) Ratio of recommendation and recommendation success rate at each turn on LastFM dataset for FacT-CRS at depth 3 and 7.

FacT-CRS to subsequently ask better questions, and make better recommendations.

4.3.1 Impact from tree depth. Figure 2 (right) shows the performance of FacT-CRS with different depths of its interaction trees. For smaller depth (at depth 3), FacT-CRS starts recommending early and more frequently. However, at the same time it compromises the success rate. For larger depth (depth 7), FacT-CRS asks more questions before making a recommendation. From the figure we see that the recommendation success rate improves by asking more questions.

4.4 Ablation Study

In this section, we evaluate the effect of each individual component of FacT-CRS by removing that component and evaluating the remaining model.

4.4.1 Impact of Candidate Items in User-item Interaction Tree. Every node in the user-item interaction tree is associated with a set of user-item pairs. If we are ready to recommend at a node in the user-item interaction tree (usually near the leaf node), we check which items are in those user-item pairs. Then we rank those items based on the predicted user-item interaction embedding. This enables us to significantly narrow down the candidate set of items. Without this component, we would have to rank all the items each time using Eq. (7) to make a recommendation. Table 3 reports our model's performance without taking into account the candidate items. The candidate item set selection design is a vital part of FacT-CRS. As Table 3 shows, this is an effective way to make our recommendations much better by narrowing down the candidate set of items.

Table 3: Effect of Candidate Items in User-item Interaction Tree.

	LastFM		BookRec		MovieLens	
	SR@10	AT	SR@10	AT	SR@10	AT
FacT-CRS	0.719	6.65	0.438	8.23	0.692	6.57
w/o Candidate	0.578	8.36	0.195	9.50	0.493	8.51

Also, the early recommendation component of FacT-CRS relies on the size of the candidate items to decide when to recommend.

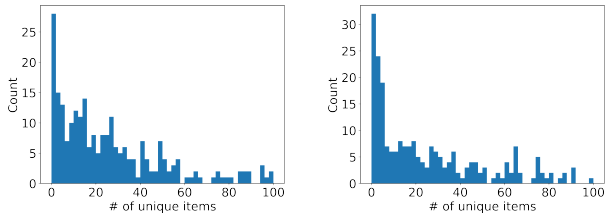

Figure 3: Histogram of the number of items in leaf nodes in a user-item interaction tree for (left) LastFM and (right) BookRec datasets.

To understand why using the items in the leaf nodes of the user-item interaction tree is important, we plot the histogram of the number of unique items in the leaf node of a user-item interaction tree. Figure 3 shows the results for LastFM and BookRec datasets. As we can see, most of the leaf nodes have very few (< 20) items. Hence, the leaf nodes work well to cluster and narrow down the correct candidate list of items. This also empirically verifies our initial assumption that using the shared attributes to cluster the user-item interactions and subsequently learning the embeddings enhance the quality of the embeddings.

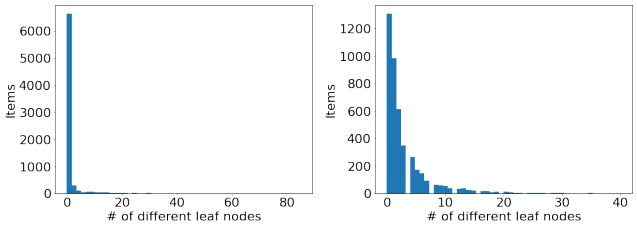

Figure 4: Histogram of the number of different leaf nodes each item appears in the user-item interaction tree for (left) Last FM and (right) BookRec datasets.

On the other hand, we also check how scattered each item is among the user-item interaction tree by recording how many different leaf nodes contain the same item. Figure 4 shows the histogram of the number of leaf nodes each item is spread across. This figure demonstrates that the locations of most items are quite concentrated. By combining the findings from Figure 3 and 4, we find that using the shared attributes to group items according to user-item interactions, FacT-CRS can effectively find a good subset of candidate items containing a small number of items.

Table 4: Ablation study of main components of FacT-CRS.

	LastFM		BookRec		MovieLens	
	SR@10	AT	SR@10	AT	SR@10	AT
FacT-CRS	0.719	6.65	0.438	8.23	0.692	6.57
¬ RF	0.349	8.30	0.117	9.52	0.223	8.73
¬ EarlyRec	0.410	9.68	0.201	9.81	0.436	9.68
¬ OnlineFeed	0.704	6.58	0.350	8.46	0.595	6.53

4.4.2 Impact of Random Forest (RF). Random forest provides a key feature of multi-turn CRS by allowing us to ask more questions after encountering a rejection. Without RF, the number of questions we can ask at most is the maximum depth of the tree H_{max}. We evaluate the significance of RF, by evaluating a single interaction tree built from the complete attribute set $\mathcal{F}$. Table 4 shows that it contributes the most among all components. By allowing FacT-CRS to ask an arbitrary number of questions and to recommend multiple times, this component makes the user-item interaction tree suitable for multi-turn CRS setting.

4.4.3 Impact of Early Recommendation (EarlyRec). For RQ2, we study the contribution of the EarlyRec strategy. To minimize the number of interactions with the user, we make an early recommendation if the candidate set of items at the current node is small enough. As we can see from Table 4, EarlyRec has significant impact on minimizing the average turn (AT). By recommending early, this component also helps the agent understand if it is on the right track, and allows the agent to make necessary corrections that improve future recommendations.

4.4.4 Impact of Handling Online Negative Feedback (OnlineFeed). To answer RQ3, we study the effectiveness of our online feedback method. The online feedback component updates the current inferred interaction embedding, when the user rejects recommendations made by the agent. As we can see from Table 4, this strategy contributes to better recommendation quality. OnlineFeed component first tries to identify the items responsible for the rejected recommendations, and corrects the predicted user-item interaction embedding to move towards the potential set of items that contains the target item. This allows FacT-CRS to choose the next interaction tree more efficiently.

4.5 Case Study

We performed the following case studies to analyze the performance of our model and to identify where we can further improve.

4.5.1 Failed Conversations. We paid special attention to the failed conversations to get a better understanding of why a conversation fails. On all three datasets, we report the average number of mentioned attributes in the failed interaction and compare it to successful interactions. Table 5 summarizes the mean and standard deviation of this results. As we can see, the average number of mentioned attributes in the failed conversations is smaller than the average number of mentioned attributes in the corresponding complete datasets.

We next look at the conversations FacT-CRS failed where the interaction contained at least p_n number of attributes in Table 6.

Table 5: Mean μ and standard deviation σ of number of mentioned attributes in the interaction for successful, failed, and all conversations.

	LastFM		BookRec		MovieLens	
	μ	σ	μ	σ	μ	σ
Successful	5.65	1.13	5.60	1.09	4.37	1.01
Failed	5.15	1.16	5.06	0.99	4.02	0.91
All	5.51	1.15	5.30	1.07	4.26	1.00

Table 6: SR@10 of FacT-CRS when at least p_n number of attributes confirmed in the interaction.

Min # attributes p_n	LastFM	BookRec	MovieLens
3	0.721	0.438	0.729
4	0.751	0.496	0.765
5	0.783	0.573	0.830
6	0.826	0.644	0.882

Our experiments show that for greater values of p_n, it becomes increasingly likely that FacT-CRS can successfully recommend the target item. This gives us the basic insight of why a conversation fails. Since the attributes mentioned in the failed conversations are very few, our model can not find sufficient information to infer which particular item the user is looking for.

4.5.2 Identified Attributes. Figure 5 shows the success rate of conversations with different interaction length p_n and the number of attributes p_k identified by FacT-CRS. Note that $p_k > p_n$ is not possible, i.e., FacT-CRS cannot identify more attributes than the total number of attributes associated with an interaction. Also, $p_k < T$, since any CRS agent can ask at most $T-1$ questions. When $p_k \leq p_n$, the white cells in Figure 5 refer to the events that are possible but did not occur. For example, on the BookRec dataset, when the number of attributes in an interaction is 9 (i.e., $p_n = 9$), FacT-CRS always identified at least 3 attributes ($p_k \geq 3$). When more attributes are identified (left to right in Figure 5), it is more likely that the conversation will be successful. Similarly, when the user mentions more attributes in an interaction (top to bottom), FacT-CRS is likely to identify more attributes and subsequently the conversations are more likely to be successful.

Our solution provides good predictive accuracy as well as minimizes the inputs required from the user. Our model outperforms the existing RL-based baselines on LastFM, BookRec and MovieLens datasets, which demonstrates the robustness of our model.

5 CONCLUSION

Multi-turn CRS is a dynamic approach to elicit the current user preference by asking a series of questions and making recommendations accordingly. Existing approaches in conversational recommender system rely heavily on reinforcement learning based policy learning, whose performance however strongly depends on the amount of training data.

In this paper, we proposed an alternative to the reinforcement learning methods and demonstrated multi-turn CRS are addressable by decision trees. To generalize a decision tree for multi-turn CRS,

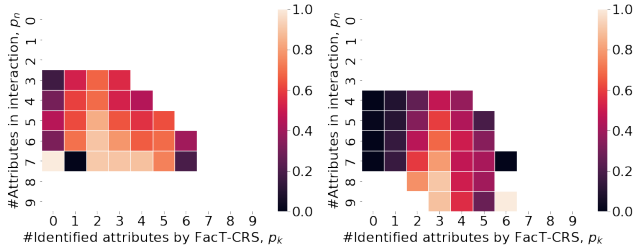

Figure 5: SR@10 of the number of attributes identified (correctly asked) by FacT-CRS for different interaction length on (left) LastFM and (right) BookRec datasets.

we addressed four key challenges in multi-turn CRS: *which questions to ask, how to rank the items, when to recommend, and how to handle the user's rejection.* We proposed building an user-item interaction tree that is able to identify different description of a certain item. The interaction tree naturally provides a way to ask questions. To effectively rank the items, we learned the embeddings of user-item interactions. For this purpose, we used a decision tree based method called the factorization tree [26], which allows us to narrow down the candidate set of items by asking questions. By leveraging the random forest, we extended factorization tree to multi-turn CRS. We solved the challenge of when to recommend, by recommending when the candidate item set is small enough. Making corrections to the interaction embedding after encountering a rejection enables us to effectively handle the users' online rejection. We extensively experimented on three benchmark CRS datasets, and compared FacT-CRS's performance with existing RL-based state-of-the-art solutions. The experimental results demonstrate that FacT-CRS outperforms on all three datasets by successfully asking questions and identifying the target items in fewer number of turns.

Our exploration in FacT-CRS sheds light on simple alternatives for multi-turn CRS. Though effective in our extensive evaluations, our solution still contains several empirically set hyper-parameters, such as the tree depth and number of trees. It is important for us to eliminate such hyper-parameters via automated tuning. In addition, currently we handle the conversations independently, even if they were from the same user. As our future work, it is important for us to study how to leverage observations from the same user or about the same item to further facilitate the conversation and recommendation.

6 ACKNOWLEDGEMENT

We thank the anonymous reviewers for their valuable comments. This work is partially supported by NSF under grant IIS-2128019, IIS-2007492 and IIS-1553568.

REFERENCES

[1] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. (2011).
[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[3] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-Scale Interactive Recommendation with Tree-Structured Policy Gradient. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 3312–3320. https://doi.org/10.1609/aaai.v33i01.33013312

[4] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. 2018. Q&R: A two-stage approach toward interactive recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 139–148.

[5] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 815–824.

[6] Zhendong Chu, Hongning Wang, Yun Xiao, Bo Long, and Lingfei Wu. 2022. Meta Policy Learning for Cold-Start Conversational Recommendation. *arXiv preprint arXiv:2205.11788* (2022).

[7] Jennifer Chu-Carroll and Michael K Brown. 1998. An evidential model for tracking initiative in collaborative dialogue interactions. In *Computational Models of Mixed-Initiative Interaction*. Springer, 49–87.

[8] 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).

[9] Corrado Gini. 1921. Measurement of inequality of incomes. *The economic journal* 31, 121 (1921), 124–126.

[10] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[11] Chen He, Denis Parra, and Katrien Verbert. 2016. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications* 56 (2016), 9–27. https://doi.org/10.1016/j.eswa.2016.02.013

[12] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 355–364.

[13] 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 International Conference on Web Search and Data Mining*. 304–312.

[14] 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 International Conference on Knowledge Discovery & Data Mining*. 2073–2083.

[15] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. *Advances in neural information processing systems* 31 (2018).

[16] 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 Transactions on Information Systems (TOIS)* 39, 4 (2021), 1–29.

[17] Kevin McCarthy, Yasser Salem, and Barry Smyth. 2010. Experience-based critiquing: Reusing critiquing experiences to improve conversational recommendation. In *International Conference on Case-Based Reasoning*. Springer, 480–494.

[18] Phuong T Nguyen, Juri Di Rocco, and Davide Di Ruscio. 2019. Building information systems using collaborative-filtering recommendation techniques. In *International Conference on Advanced Information Systems Engineering*. Springer, 214–226.

[19] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. 2019. Value-aware recommendation based on reinforcement profit maximization. In *The World Wide Web Conference*. 3123–3129.

[20] Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*. 117–126.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[22] Gerard Salton. 1971. *The SMART retrieval system—experiments in automatic document processing*. Prentice-Hall, Inc.

[23] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[24] Barry Smyth and Lorraine McGinty. 2003. An analysis of feedback strategies in conversational recommenders. In *the Fourteenth Irish Artificial Intelligence and Cognitive Science Conference (AICS 2003)*. Citeseer.

[25] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval*. 235–244.

[26] Yiyi Tao, Yiling Jia, Nan Wang, and Hongning Wang. 2019. The FacT: Taming latent factor models for explainability with factorization trees. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 295–304.

[27] Frederich N Tou, Michael D Williams, Richard Fikes, D Austin Henderson Jr, and Thomas W Malone. 1982. RABBIT: An Intelligent Database Assistant.. In *AAAI*. 314–318.

[28] Amos Tversky and Itamar Simonson. 1993. Context-dependent preferences. *Management science* 39, 10 (1993), 1179–1189.

[29] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 209–218.

[30] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[31] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5329–5336.

[32] Ji Wu, Miao Li, and Chin-Hui Lee. 2015. A probabilistic framework for representing dialog systems and entropy-based dialog management through dynamic stochastic state evolution. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 11 (2015), 2026–2035.

[33] Wenquan Wu, Zhen Guo, Xiangyang Zhou, Hua Wu, Xiyuan Zhang, Rongzhong Lian, and Haifeng Wang. 2019. Proactive Human-Machine Conversation with Explicit Conversation Goal. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3794–3804. https://doi.org/10.18653/v1/P19-1369

[34] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 931–940.

[35] 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 international conference on web search and data mining*. 364–372.

[36] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proc. IEEE* 101, 5 (2013), 1160–1179.

[37] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, and Changyou Chen. 2019. Text-based interactive recommendation via constraint-augmented reinforcement learning. *Advances in neural information processing systems* 32 (2019).

[38] 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.

[39] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th acm international conference on information and knowledge management*. 177–186.

[40] 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.

[41] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 315–324.

[42] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 179–188.

[43] Jie Zou, Yifan Chen, and Evangelos Kanoulas. 2020. Towards question-based recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 881–890.

[44] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 816–824.