Enhancing Conversational Recommendation Systems with Representation Fusion

YINGXU WANG*, School of Computer Science and Engineering, Sun Yat-sen University, China and Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates
XIAORU CHEN*, School of Computer Science and Engineering, Sun Yat-sen University, China
JINYUAN FANG, School of Computing Science, University of Glasgow, UK
ZAIQIAO MENG, School of Computing Science, University of Glasgow, UK
SHANGSONG LIANG†, School of Computer Science and Engineering, Sun Yat-sen University, China and Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates

Conversational Recommendation Systems (CRSs) aim to improve recommendation performance by utilizing information from a conversation session. A CRS first constructs questions and then asks users for their feedback in each conversation session in order to refine better recommendation lists to users. The key design of CRS is to construct proper questions and obtain users' feedback in response to these questions so as to effectively capture user preferences. Many CRS works have been proposed; however, they suffer from defects when constructing questions for users to answer: (1) employing a dialogue policy agent for constructing questions is one of the most common choices in CRS, but it needs to be trained with huge corpus; (2) it is not appropriate that constructing questions from a single policy (e.g., A CRS only selects attributes that the user has interacted with) for all users with different preferences. To address these defects, we propose a novel CRS model, namely, a **R**epresentation **F**usion-based **C**onversational **R**ecommendation model (RFCR), where the whole conversation session is divided into two subsessions (i.e., Local Question Search subsession and Global Question Search subsession) and two different question search methods are proposed to construct questions in the corresponding subsessions without employing policy agents. In particular, in the Local Question Search subsession we adopt a novel graph mining method to find questions, where the paths in the graph between users and attributes can eliminate irrelevant attributes; in the Global Question Search subsession we propose to initialize user preference on items with the user and all item historical rating records and construct questions based on user's preference. Then, we update the embeddings independently over the two subsessions according to user's feedback and fuse the final embeddings from the two subsessions for the

*Both authors contributed equally to the paper.
†Corresponding author

Authors' addresses: Yingxu Wang, School of Computer Science and Engineering, Sun Yat-sen University, No.132, EastWaihuan Road, Guangzhou, China, 510006, Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, AI Diyafah St, Abu Dhabi, United Arab Emirates, 7909, Yingxu.Wang@mbzuai.ac.ae; Xiaoru Chen, School of Computer Science and Engineering, Sun Yat-sen University, No.132, EastWaihuan Road, Guangzhou, China, 510006, chenxr27@mail2.sysu.edu.cn; Jinyuan Fang, School of Computing Science, University of Glasgow, Glasgow, UK, fangjy6@gmail.com; Zaiqiao Meng, School of Computing Science, University of Glasgow, Glasgow, UK, zm324@cam.ac.uk; Shangsong Liang, School of Computer Science and Engineering, Sun Yat-sen University, No.132, EastWaihuan Road, Guangzhou, China, 510006, Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, AI Diyafah St, Abu Dhabi, United Arab Emirates, 7909, liangshangsong@gmail.com.

1559-1131/2022/3-ART $15.00
https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

recommendation. Experiments on three real-world recommendation datasets demonstrate that our proposed method outperforms five state-of-the-art baselines.

CCS Concepts: • **Information systems → Recommender systems**.

Additional Key Words and Phrases: Conversational Recommendation Systems; Interactive Recommendation; Representation Learning

ACM Reference Format:
Yingxu Wang, Xiaoru Chen, Jinyuan Fang, Zaiqiao Meng, and Shangsong Liang. 2022. Enhancing Conversational Recommendation Systems with Representation Fusion. *ACM Trans. Web* 1, 1 (March 2022), 34 pages. https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

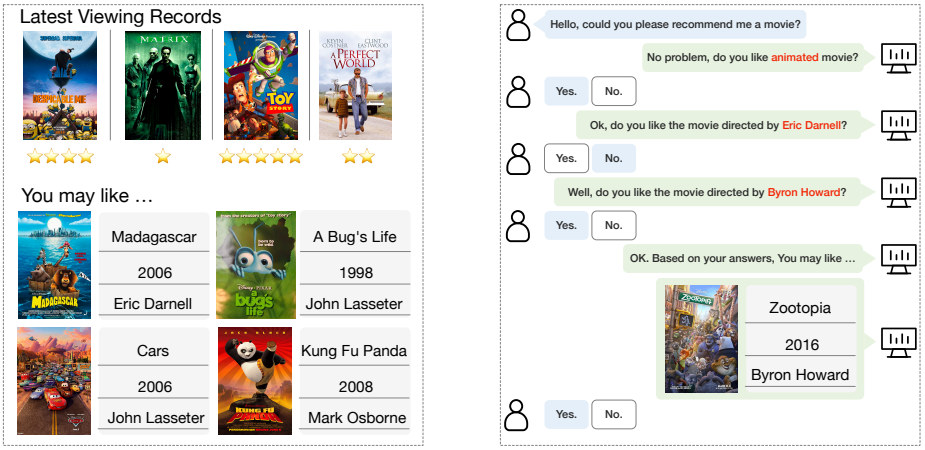
1 INTRODUCTION

Recommendation systems have been widely adopted in E-commerce platforms (e.g., Amazon [1] and Taobao [2]) to help users find their desired items. As shown in Figure 1.(a), traditional recommendation systems recommend items by inferring user preference based on historic user-item ratings [18, 27, 33, 42, 51, 58]. However, the rating data is usually static and therefore challenging to be utilized to represent users' dynamic interests over time. This challenge hinders recommendation systems to make effective and accurate recommendations. To better model users' interest evolution and improve recommendation performance, recent research has focused on Conversational Recommendation System (CRS) [10, 15, 19, 21]. CRS aims to infer user interests by asking a few questions regarding user preference and then produces relevant recommendations by utilizing the responses [29, 35, 55]. As a result, compared with traditional recommendation systems, CRS can better infer users' current purchasing intention and generate high-quality recommendations by utilizing users' feedback.

There are multiple interaction paradigms for CRS, which focus on different aspects of CRS such as exploitation-exploration (EE) trade-offs for cold users [36, 53, 66], question-driven approaches [1, 62, 67], multi-turn conversational recommendation strategy [30, 31, 49, 61], and dialogue understanding and generation [24, 32, 44]. In this paper, we are consistently focusing on the multi-turn conversation recommendation paradigm, where the system makes recommendations for the user in one conversation session [9, 30]. During this conversation session, the system may ask the user questions multiple times and collect his/her feedback. Afterward, the user preference can be inferred from the feedback and leveraged to make appropriate recommendations at the end of the interactions. One example of such a conversation recommendation paradigm is shown in Figure 1.(b). In this example, a CRS tries to recommend movies to the user. To achieve this purpose, a CRS proactively inquires about the user's preference for movies with a series of questions and then waits for the user's responses. After receiving the user's feedback in response to these questions, a CRS infers the user's current interest in movies and recommends the most suitable movie for the user.

Question construction in the conversation session to effectively infer the user's current interests is the key to the success in CRS. These questions are supposed to be as few as possible while being able to be utilized to represent the user's potential interests. Many question construction policies have been proposed to construct questions [8, 40, 69]. However, these methods suffer from the following defects: (1) Existing question construction methods propose to construct questions either by exploitation or exploration. Specifically, methods from the perspective of exploitation propose to construct questions from the attributes of items that the user has interacted with [30, 61]. This kind of method can effectively infer users' past interests. However, it is challenging for these exploitative methods to explore users' potential new interests. In contrast, methods from the perspective of

[1] www.amazom.com
[2] www.taobao.com



(a) Traditional Recommendation System (b) Conversational Recommendation System

Fig. 1. A comparison between a traditional recommendation system and a CRS for movie recommendation. (a) A traditional recommendation system, which consists of one user's latest viewing records and a movie list recommended for this user; (b) A CRS, where one user is asked some questions and then provides his/her responses one-by-one and finally receives a movie recommended by a CRS. The terms in red shown in (b) represent the asked attributes (keywords in questions).

exploration construct questions from all the attributes in the datasets [31, 49]. This kind of method can discover users' potential new interests. However, to achieve this purpose, these exploratory methods often require asking lots of questions. This is due to the fact that there is a large number of attributes in the datasets, causing these methods to construct some irrelevant questions regarding the user's current interests. (2) In order to construct questions for the user, most CRS methods employ a *policy agent* to guide the construction of questions [5, 50, 67]. However, the training of these policy agents requires a huge and comprehensive corpus so that the learned agents can provide an accurate policy under all circumstances. Meanwhile, it is also computationally expensive to learn policy agents from such a huge corpus.

To address the aforementioned defects, we propose a novel CRS model, referred to as the **R**epresentation **F**usion-based **C**onversational **R**ecommendation (RFCR) method. Since constructing questions from the exploitative strategy and the exploratory strategy alone have their strengths and drawbacks, we propose a fusion method to take advantage of both strategies while avoiding their shortcomings. Specifically, we divide the whole conversation session into two subsessions, namely, the Local Question Search (LQS) subsession and the Global Question Search (LQS) subsession, as shown in Figure 2. In the LQS subsession, we construct questions from the strategy of exploitation, i.e., by leveraging attributes of the items that the user has interacted with. In the GQS subsession, we construct questions from the strategy of exploration, i.e., by leveraging all the attributes in the dataset. We would like to stress that questions from these two strategies are complementary to each other. Therefore, by fusing information from these two types of questions, we can discover the user's past interests as well as explore the user's potential new interests. In addition, since it is difficult to learn an effective policy agent to guide the construction of questions, we do not employ such policy agents in both LQS and GQS subsessions. Instead, we propose to guide the construction of questions in these two subsessions using the statistics calculated from the dataset, which avoids the need for a large and comprehensive corpus. Moreover, we can also reduce the time and space complexity during training compared to other CRS methods with policy agents. With these two

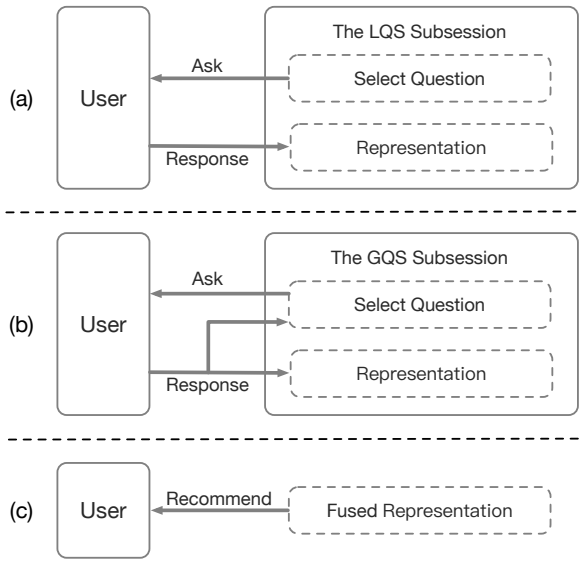


Fig. 2. Overview of our proposed RFCR model. (a) The LQS subsession, where we ask users questions and wait for responses from users; (b) The GQS subsession, where we ask users questions, receive users' responses and update the question selection policies according to users' feedback; (c) The recommendation module, where we fuse the user-item embeddings from the LQS and GQS subsessions and recommend items for users.

improvements, we can achieve a better performance in inferring the user's preference with fewer questions while being computationally efficient for learning.

Specifically, in the LQS subsession of our proposed RFCR model, we propose to construct a directed graph based on user-item and item-attribute associations. In this directed graph, nodes represent users, items, and attributes, and edges represent the connections from users to items and the connections from items to attributes. An example of such a directed graph is shown in Figure 4. Note that there might be several paths from the user to a specific attribute. We assume that the more paths from the user to a certain attribute, the more the corresponding attribute can reflect the user's long-term preference. Therefore, in the LQS subsession, we propose to construct questions from these attributes with the most paths. In each turn of the conversation, after the user's feedback on the question is obtained, the user and item embeddings are updated accordingly. In the GQS subsession of our proposed RFCR model, we maintain a user belief over the user's preference for all the items in the dataset and constantly update this belief with the feedback from the user. Specifically, we model the user's preference for all the items using a (multinomial) probability distribution, with each component representing the probability that the user intends to purchase this item. In each turn of the conversation, we search from the attribute set based on this probability distribution and then construct questions for users with selected attributes. Similar to the LQS subsession, the user and item embeddings are also updated based on the user's feedback on the question. It is worth noting that we update the user and item embeddings independently in the LQS and GQS subsessions. To effectively leverage information from these two subsessions, we further propose a representation fusion method to fuse embeddings obtained from the LQS and GQS subsessions. We believe that embeddings obtained from the LQS subsession could represent the user's long-term interests and the embeddings from the GQS subsession could represent the user's potential new interests. Therefore, by fusing these two kinds of embeddings, we can obtain a more precise representation to accurately infer the user's current preference.

To the best of our knowledge, constructing questions from both the exploitative and the exploratory strategies without policy agents have not been explored before in the CRS literature. To demonstrate the effectiveness of RFCR model, we evaluate its performance on three real-world datasets. We aim at answering these research questions in the experiments: (1) What is the impact of the splitting schemes of Q on the performance of our RFCR model? (2) How does our RFCR model perform compared with existing conversational recommendation methods? (3) What is the impact of the total number of turns in conversation on the performance of RFCR? (4) What are the effects of key components in CRS, i.e., LQS, GQS? (5) How does our RFCR perform on the cold-start problem? (6) How interpretable are the questions for users asked by our RFCR model?

Experimental results show that compared with baselines, our RFCR model could achieve better performance with fewer turns, i.e., asking fewer questions. In summary, our contributions include:

- We propose a novel framework for CRS, where the questions are constructed from both the exploitative and the exploratory strategies. This is achieved by dividing the whole conversation session into two subsessions and fusing these embeddings from these two subsessions to make recommendations.
- We propose a novel CRS model, RFCR. Our RFCR model achieves better performance with fewer turns and without employing policy agents, which improves the performance of CRS and releases the heavy burden of training policy agents with a large corpus.
- We conduct experiments in three real-world datasets. The experimental results on these datasets show that our RFCR model can achieve superior performance with fewer turns and outperform the state-of-the-art baselines, which demonstrates the effectiveness of the proposed RFCR method.

We organize the remainder of the paper as follows. § 2 reviews related work; § 3 presents the problem formalization; § 4 introduces the methodology; § 5 presents the experimental setting; and § 6 presents the experimental results. Finally, we conclude the paper and discuss future work in § 7.

2 RELATED WORK

In this section, we briefly describe two lines of research that are closely related to our work, namely, traditional recommendation systems and CRS.

2.1 Traditional Recommendation Systems

Traditional recommendation systems, such as Matrix Factorization (MF) [27], Factorization Machine (FM) [51], estimate the affinity scores between users and items by learning user preferences through historic user-item log data, e.g., click history, visit log, ratings on items. To improve the performance of these methods, some deep neural network-based recommendation methods have been proposed. For example, Neural MF [16], DeepFM [18] and PDMF [63] leverage neural networks [3, 6] to infer users' preferences from historic user-item interactions based on the collaborative filtering hypothesis [52]. These methods have achieved outperformance and have been deployed in many real-world scenarios [11, 13, 71]. Recently, graph-based models have been proposed to enhance neural network-based recommendation models [43, 59, 70]. They leverage graph-based methods to model complex relations among users, items, and attributes, which is beneficial to better capturing the affinities among entities from historic data.

However, these traditional recommendation models suffer from intrinsic weaknesses: (1) they assume that the users' interests in items are static over time, which causes these traditional recommendation models cannot capture the users' dynamic preferences [37, 39]. (2) Since there are no channels for these traditional recommendation models to interact with users, it is hard

to provide accurate reasons to explain users' special purchasing intentions on certain items [28]. These limitations motivate new research on recommendation systems. Later work tries to introduce Markov models [25, 54] and bandit models [12, 56, 60] to incorporate dynamic information. Markov models assume that the next action is conditioned on only the previous action in a sequential list and have been successfully adopted to characterize short-range item transitions for the recommendation. Bandit models aim to maximize the accumulated reward for playing arms during a period of time by solving the exploit-explore problem. However, both Markov models and bandit models cannot interact with users to incorporate their feedback and produce more precise user-item representations. In addition, bandit models only can work in small arm pools, where the large attribute pool makes bandit models hard to efficiently hit the user-preferred attributes duration exploration [34]. So, the performance of these models still remains to be unsatisfactory.

However, our RFCR model has the ability to interact with users in one conversation session, then receive responses from them and dynamically update users' preferences over items based on their feedback. Moreover, our RFCR model is able to construct questions for users from the exploitative strategy and the exploratory strategy to capture their potential needs and intrinsic intentions from a large attribute pool.

2.2 Conversational Recommendation System

A CRS is more effective in addressing the above intrinsic defects than traditional recommendation systems. Specifically, in order to capture users' dynamic preferences [38], a CRS introduces a task-oriented conversation session to ask users questions and collect their feedback [9]. The feedback from users may consist of their direct or indirect descriptions of answers to the questions asked by a CRS, with which a CRS can infer users' true purchase preferences and produce more accurate recommendation lists [29].

Motivated by the potential in interacting with users and inferring users' dynamic preferences from their real-time feedback of conversational recommendation systems, many research efforts have been devoted to exploring this topic. Various task formulations have been deployed, where CRSs focus on different aspects under distinct hypotheses and application scenarios. According to the different research on CRSs, Lei et al. [28] categorize CRSs into four directions: (1) EE trade-offs for cold users, (2) question-driven approaches, (3) multi-turn conversational recommendation strategy, (4) dialogue understanding and generation.

- Direction (1) focuses on applying bandit-based models to CRS in order to balance EE trade-offs for cold-start users [2, 36]. Christakopoulou et al. [9] first propose to employ bandit algorithms in CRS to help users better find their desired restaurants. Zhou et al. [68] incorporate users' attitudes towards music in the recommendation to tackle the fast-changing music preferences. Zhang et al. [66] design a new UCB algorithm to address the key-terms selections and consider the users' feedback on items in CRSs. Li et al. [36] propose to seamlessly unify attributes and items in the same arm space and achieve the EE trade-offs automatically using the framework of Thompson Sampling.
- In direction (2), the key research issue is to estimate the users' preferences towards attributes. Question-driven CRSs construct questions from an attributes pool, and they assume it is effective to infer a user's preference for items by asking the user whether she/he likes certain attributes and all users would accept the recommendation lists generated from a CRS [8]. Bi et al. [5] propose to identify users' intention by collecting negative feedback on attributes of items. Sun and Zhang [55] propose to ask users about his/her preference on certain items, and users specify values on these items. Zou et al. [74] propose to ask user's feedback towards attributes and predict items by matching attribute query and item description.

- Direction (3) focuses on the multi-turn CRS, where the system produces a recommendation list by asking several turns of questions in one conversation session. The goal of a multi-turn CRS is to accurately inquire about users' preferences by asking as few questions as possible, i.e., with fewer turns. To this end, Lei et al. [30] estimate the importance of each attribute with information entropy and then utilize a policy agent to select a suitable attribute from the attribute pool in each conversation turn. Lei et al. [31] incorporate path searching on knowledge-graph (KG) into CRS and formulate a series of questions to collect the user's response. Xu et al. [61] propose to incorporate both attribute-level feedback and item-level feedback from users in a CRS. Ren et al. [49] model a user's preference in a finer granularity by identifying the most relevant relations from a structured knowledge graph.
- The core problem of direction (4) is natural language understanding and generation. Direction (4) focuses on how to understand the user's preferences and purchasing intentions from the user's feedback, generate fluent responses, and provide natural and effective dialogue actions. Li et al. [35] focus on natural language generation and estimate user's preference based on entities mentioned in the dialogue. Pei et al. [44] propose a novel memory network to gradually enrich user profiles as dialogues progress and to simultaneously improve response selection based on the enriched profiles. Li et al. [32] focus on medical dialogue response generation with a large-scale unlabeled corpus and design a generative model to model unobserved patient state and physician actions.

Although most of the above models have improved the performance of many existing CRSs, they still suffer from two defects. (1) Existing question construction methods propose to construct questions from either the exploitative strategy or the exploitative strategy, which causes that a CRS cannot discover the user's past interests as well as explore the user's potential new interests. (2) In addition, it is difficult to learn an effective policy agent to guide the construction of questions for most CRS models, as the training of these policy agents requires a huge and comprehensive corpus. Our RFCR model constructs questions for users from the exploitative strategy and the exploratory strategy and proposes a fusion method to take advantage of both strategies. In addition, Our RFCR model utilizes the statistics calculated from the dataset to guide the construction of questions instead of employing policy agents, which avoids the need for a large and comprehensive corpus.

3 PRELIMINARIES

In this section we first introduce the main notations used across the whole paper, and then formulate the task to be addressed in the paper.

3.1 Notations

We use $\mathcal{U} = \{u_1, u_2 \cdots u_M\}$ and $\mathcal{V} = \{v_1, v_2 \cdots v_N\}$ to denote the sets of users and items respectively, where M is the number of users and N is the number of items. Let $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M] \in \mathbb{R}^{M \times D}$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N] \in \mathbb{R}^{N \times D}$ be the embedding matrices of users and items respectively, with $\mathbf{u}_i$ being the embedding of user u_i, $\mathbf{v}_j$ being the embedding of item v_j, and D being the size of the dimension. Each of the items is profiled by multiple binary attributes from an attribute set $\mathcal{A} = \{a_1, a_2 \cdots a_L\}$ with L being the number of attributes. For the sake of convenience, we use $\mathcal{A}_j$ to denote the set of attributes associated with v_j. For example, the attribute set of item v_j can be represented as $\mathcal{A}_j$ ={*Apple, red case, 6.1 inches, 5G communication, 4GB RAM*}. Let $\mathbf{R} \in \mathbb{R}^{M \times N}$ be the user-item rating matrix, with each component representing the rating that the user gives to the item. We use $\mathbf{R}_i$ to represent the i-th row of $\mathbf{R}$ and $\mathbf{R}_{\_j}$ to represent the j-th column of $\mathbf{R}$. Let $\mathbf{Y} \in \mathbb{R}^{M \times N}$ be the online affinity matrix for collecting users' feedback, where $\mathbf{Y}_i$ represents the i-th

Table 1. Main notations used in this paper.

Notations	Descriptions
M	The number of users
N	The number of items
L	The number of attributes
D	The dimensions of embeddings
$\mathcal{U} = \{u_i\}_{i=1}^{M}$	The set of users
$\mathcal{V} = \{v_i\}_{i=1}^{N}$	The set of items
$\mathcal{A} = \{a_i\}_{i=1}^{L}$	The set of attributes
$\mathcal{A}_j$	The set of attributes contained by item v_j
$\mathbf{R} \in \mathbb{R}^{M \times N}$	The user-item rating matrix
$\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^{M}$	Latent representation matrix for all the users
$\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N}$	Latent representation matrix for all the items
$\mathbf{Y} \in \mathbb{R}^{M \times N}$	Online affinity matrix
$r^*_{u_i}$	The generating rating of user u_i's from his/her historic purchasing propensity rating
$r^*_{v_j}$	The generating rating of item v_j from its historic purchasing propensity rating
$\boldsymbol{\pi}^*$	Preference distribution over items in the GQS subsession
$\mathbf{P}$	Prior belief over $\boldsymbol{\pi}^*$ in the GQS subsession
Q	The total number of turns in an conversation session
Q_1	The number of turns in the LQS subsession
Q_2	The number of turns in the GQS subsession

row of $\mathbf{Y}$ and $\mathbf{Y}_{\cdot j}$ represents the j-th column of $\mathbf{Y}$. The main notations used across the whole paper are summarized in Table 1.

3.2 Problem Formalization

Following [8, 45, 55, 74], we consider one trial of asking a question and answering the question as a turn and set conversational recommendation as a multi-turn scenario, where a CRS provides a conversation session to ask users questions multiple times and recommends items once it reaches the end of conversation session. In addition, a CRS model normally consists of two modules, namely the offline representation learning module and the online updating module. The goal of the offline representation learning module is to learn the initial embeddings of users and items based on the historic user-item rating matrix. Afterwards, in the online updating module, these initial embeddings are updated based on the users' online feedback on questions asked by the CRS to better capture users' current preferences.

In this paper, a CRS starts with an arbitrary user u_i. Then, in the conversational session, a CRS would select an attribute a from the attribute set and ask u_i whether she/he likes this particular attribute a. Finally, user u_i would respond with "yes" or "no". [3] Note that when constructing questions from attributes, a CRS follows the form-based setting[20], where a CRS constructs pre-defined format questions for users instead of dynamically generated natural languages. For example, u_i may be asked a question "Are you seeking for a *red color* item?". If u_i likes items with this attribute, then u_i would respond with "yes" and otherwise "no". It is also possible to compose questions to elicit an enumerated response, i.e., "Which movie genre would you consider? I have action, drama ..." However, following previous works [36, 61], we only focus on binary-attributed questions in this task. In each turn t ($t = 1, 2, ...T$; T denotes the number of turns in conversation session),

[3] Any response to the questions is possible, but to be aligned with the previous works [30, 31] and alleviate the effort of conversation, we make the response as simple as "yes" or "no".

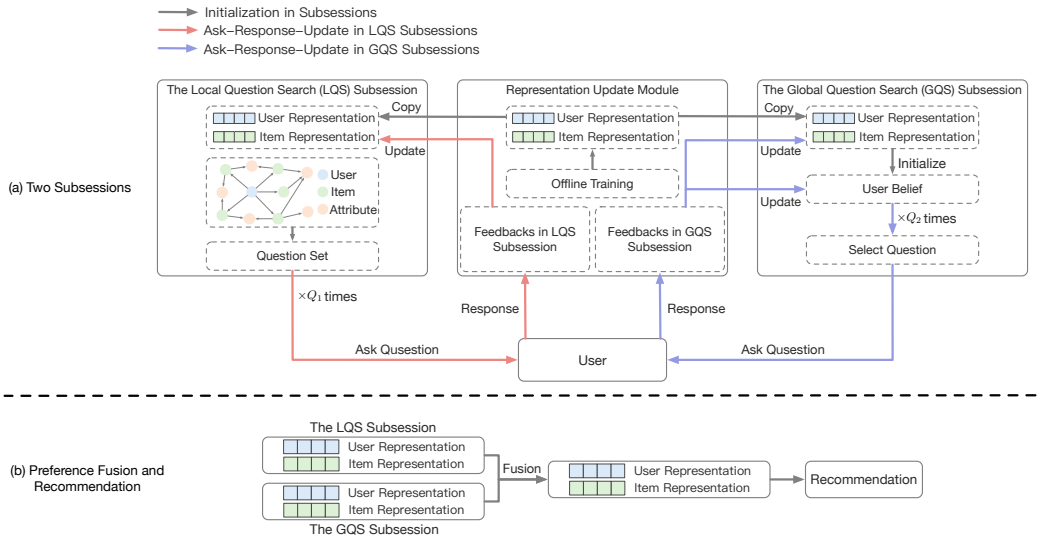


Fig. 3. The framework of RFCR. Subfigures from the top to the bottom: (a) First, we get the initial user-item representations from the offline training module when the user enters our RCFR model. Then the user would interact with the system in the Local Question Search (LQS) subsession, and the system would update the user-item representations in the LQS subsession according to the update method in the representation update module. Then the user would interact with the system in the Global Question Search (GQS) subsession, and the system would update the user-item representations in the representation update module. (b) After the two stages, we fuse the two sets of representations and make recommendations. An example of a whole conversation session can refer to Figure 1.(b).

a CRS needs to repeat the above process. At the end of conversation session, a CRS produces a recommendation list for user u_i. Based on the above setting, we use the user-item rating matrix $\mathbf{R}$ and the attribute set $\mathcal{A}$ as inputs and retrieve the users' desired items as outputs. Furthermore, we set the major goal of our model to produce desired recommendations within as few turns as possible and assume users are willing and active to participate in our CRS.

4 THE MODEL

In this section, we detail our **R**epresentation **F**usion-based **C**onversational **R**ecommendation (RFCR) model. In particular, in § 4.1, we provide an overview of our proposed RCFR model; in § 4.2, we detail how our representation method is updated in offline and online training manners, respectively; in § 4.3 and § 4.4, we propose a novel question search architecture to construct proper questions for users from the exploitative strategy and the exploratory strategy, respectively; in § 4.5, we fuse the user-item embeddings from these two subsessions to produce the final recommendation.

4.1 Overview

We provide an overview of our CRS in Figure 3. We first initialize the representations of all users and all items in the offline training module, and then a user would interact with two conversation subsessions, respectively. Once the conversation is done, RFCR aims to infer two categories of representations of users and items from the Local Question Search (LQS) subsession (§ 4.3) and the Global Question Search (GQS) subsession (§ 4.4), respectively. We then fuse and concatenate these two categories of representations to obtain the final representations of users and items. With

the final representations, we are able to retrieve a ranked list of items whose final representations match the final representation of the user.

In particular, in the LQS subsession, to represent the associations among user-item-attribute intuitively, we first utilize the original user-item rating matrix $\mathbf{R}$ and the attribute set $\mathcal{A}$ to construct a directed graph, where nodes represent users, items, and attributes, and edges represent the connections from users to items and from items to attributes. With this directed graph, we rank the attributes according to the number of paths from the user to attributes and select the attributes that appear more frequently in the paths to form a problem set. A question would be then selected from the question set to display and wait for the user's response/answer as feedback. Once the user provides the answer in response to the question, the Representation Update Module is then leveraged to update the representations of users and items in the LQS subsection by taking into account the information from both the question set and the answers. After selecting the attributes for constructing questions in the LQS subsession, we will filter these attributes from the entire attribute pool and assign the rest to the GQS subsession in order to avoid the repeated selection of the same attributes. In the GQS subsession, to obtain the user's initial preference on items, we first complement the rating record of the user based on both the user's historic ratings and all historic ratings of items. Then, we maintain a user belief over the user's preference for all the items and constantly update this belief with the feedback from the user. Specifically, we model the user's preference for all the items using a (multinomial) probability distribution, with each component representing the probability that the user intends to purchase this item. In each turn of the conversation, we search from the attribute set $\mathcal{A}$ based on this probability distribution and then ask the user one question with the selected attribute. Similar to the LQS subsession, the user and item embeddings are also updated based on the user's feedback on the questions in the GQS subsession.

For convenient discussion, let Q be the total number of question-response turns in a conversation session, denoting our model selects Q questions in total. Let Q_1 be the number of question-response turns in the LQS subsession, denoting our model selects Q_1 attributes according to the question search method in the LQS subsession. Similarly, let Q_2 be the number of question-response turns in the GQS subsession, denoting our model selects Q_2 attributes according to the question search method in the GQS subsession. Thus, according to this setting, the quota of the questions the system can form the questions is Q, and $Q = Q_1 + Q_2$. We will explore how to optimize Q_1 and Q_2 given the quota Q in § 6.1.

4.2 Representation Update Module

In this section, we detail the representation update modules in both the LQS and the GQS subsessions as shown in Figure 3. In each subsession, our RCFR model constructs questions for users according to the question search method, and users would provide their feedback in response to the question in each turn of conversation. Finally, our RCFR model updates $\mathbf{U}$ and $\mathbf{V}$ according to the feedback in each turn and the historic information.

Note that we only need to update $\mathbf{Y}_i \in \mathbb{R}^N$ corresponding to the user u_i who is currently involved in the conversation session. In addition, we use an indicator vector $\mathbf{y}^l \in \mathbb{R}^N$ to denote the l-th feedback from the l-th question, where the j-th dimension of $\mathbf{y}_j^l$ serves for the j-th item in $\mathcal{V}$:

$$\mathrm{y}_j^l = \mathbb{1}(a_j^l = a_*^l), \tag{1}$$

where a_j^l is 1 if item v_j contains the l-th asked attribute a^l, and otherwise 0. Similarly, a_*^l denotes whether the target item contains the l-th asked attribute a^l, and it is 1 if the target item contains it, and otherwise 0. $\mathbb{1}(\cdot)$ is an indicator function.

Thus, $\mathbf{Y}_i^l$ can be computed as:

$$\mathbf{Y}_i^l = \sum_{t=1}^{l} \mathbf{y}^t, \tag{2}$$

at the end of the l-th turn in the LQS subsession and the GQS subsession, where

$$\mathrm{Y}_{ij}^l = \sum_{t=1}^{l} \mathrm{y}_j^t.$$

At the end of each turn, we obtain the user's response and update $\mathbf{Y}$ according to the above procedure. Then we take into account both $\mathbf{R}$ and $\mathbf{Y}$ to update user and item embeddings, respectively. To update the representations of users and items with off-line information as well as online feedback efficiently, we choose the Question-based Matrix Factorization method (QMF) [74]. Other choices of the base methods are also possible, but we found that their performance is essentially the same. The generative process of QMF is as follows:

1. For each user u_i, $i = 1, \ldots, M$, draw a user embedding $\mathbf{u}_i \sim \mathcal{N}\left(\mathbf{u}_i \mid 0, \lambda_u^{-1}\mathbf{I}\right)$;
2. For each item v_j, $j = 1, \ldots, N$, draw a user embedding $\mathbf{v}_j \sim \mathcal{N}\left(\mathbf{v}_j \mid 0, \lambda_v^{-1}\mathbf{I}\right)$;
3. For the user-item rating score of i-th user and j-th item in $\mathbf{R}$, draw $\mathrm{R}_{ij} \sim \mathcal{N}(\mathrm{R}_{ij} \mid \mathbf{p}^\top(\mathbf{u}_i \odot \mathbf{v}_j), 1)$;
4. For the user-item feedback score of i-th user and j-th item in $\mathbf{Y}$, draw $\mathrm{Y}_{ij} \sim \mathcal{N}(\mathrm{Y}_{ij} \mid \mathbf{q}^\top(\mathbf{u}_i \odot \mathbf{v}_j), \gamma^{-1}\mathbf{I})$ for each question asked.

In the above, λ_u, λ_v are the hyper-parameters for modeling the variances of embeddings, and γ is a hyper-parameter for modeling the variance in Y_{ij}. The operator $\odot$ denotes the element-wise product. $\mathbf{p}$ and $\mathbf{q}$ are the vectors that control the weights of each dimension in $\mathbf{u}_i \odot \mathbf{v}_j$ when applying $\mathbf{u}_i$ and $\mathbf{v}_j$ to generate R_{ij} and Y_{ij}. The intuition behind this is that $\mathbf{p}$ and $\mathbf{q}$ can capture some differences when generating R_{ij} and Y_{ij}.

Following the settings in § 3.2, we use an offline representation learning module and an online updating module in our RCFR model. In what follows, we detail these two modules, respectively.

Offline training. In the offline training module, the goal is to learn the initial representations of users and items based on the historic user-item rating matrix $\mathbf{R}$, which is collected from the log data. According to the QMF method, the maximization of the posterior distributions over $\mathbf{U}$ and $\mathbf{V}$ can be reformulated as follows:

$$\max_{\mathbf{U},\mathbf{V},\mathbf{p}} p\left(\mathbf{U}, \mathbf{V}, \mathbf{p} \mid \mathbf{R}, \lambda_u, \lambda_v, \lambda_{\mathbf{p}}\right),$$

where $\lambda_{\mathbf{p}}$ is a trade-off parameter governing the effect of $\mathbf{p}$ for regularization in the model. We reformulate the above posterior probability as the minimization of its negative logarithm, which is given by:

$$\begin{aligned} & -\log p(\mathbf{U}, \mathbf{V} \mid \mathbf{R}, \mathbf{p}) \\ \propto \quad & \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} \mathbb{1}(\mathrm{R}_{ij} > 0)\left(\mathrm{R}_{ij} - \mathbf{p}^\top\left(\mathbf{u}_i \odot \mathbf{v}_j\right)\right)^2 \\ & + \sum_{i=1}^{M} \frac{\lambda_u}{2}\|\mathbf{u}_i\|_2^2 + \sum_{j=1}^{N} \frac{\lambda_v}{2}\left\|\mathbf{v}_j\right\|_2^2 + \frac{\lambda_{\mathbf{p}}}{2}\|\mathbf{p}\|_2^2, \end{aligned} \tag{3}$$

where $\mathbb{1}(\cdot)$ is an indicator function returning 1 if the term in the bracket is true, or otherwise returning 0. As a result, in the offline training module, initial $\mathbf{U}, \mathbf{V}, \mathbf{p}$ are learned given $\mathbf{R}$ according to Eq. (3).

Online updating. In the online training module, we combine the online affinity matrix $\mathbf{Y}$ and the historic user-item rating matrix $\mathbf{R}$ to update the embeddings of users and items. According to the QMF method, the maximization of the posterior distributions over $\mathbf{U}$ and $\mathbf{V}$ can be reformulated as follows:

$$\max_{\mathbf{U},\mathbf{V},\mathbf{p},\mathbf{q}} p\left(\mathbf{U},\mathbf{V},\mathbf{p},\mathbf{q} \mid \mathbf{R},\mathbf{Y},\lambda_u,\lambda_v,\lambda_{\mathbf{p}},\lambda_{\mathbf{q}},\gamma\right),$$

where $\lambda_{\mathbf{q}}$ is a trade-off parameter governing the effect of $\mathbf{q}$ for regularization in the model. Then we reformulate the above posterior probability as the minimization of its negative logarithm, which is:

$$\begin{aligned} & -\log p(\mathbf{U},\mathbf{V} \mid \mathbf{R},\mathbf{Y},\mathbf{p},\mathbf{q}) \\ \propto\ & \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N}\mathbb{1}(\mathrm{R}_{ij}>0)\left(\mathrm{R}_{ij}-\mathbf{p}^{\top}\left(\mathbf{u}_i\odot\mathbf{v}_j\right)\right)^2 \\ & +\frac{\gamma}{2}\sum_{i=1}^{M}\sum_{j=1}^{N}\mathbb{1}(\mathrm{Y}_{ij}>0)\left(\mathrm{Y}_{ij}-\mathbf{q}^{\top}\left(\mathbf{u}_i\odot\mathbf{v}_j\right)\right)^2 \\ & +\sum_{i=1}^{M}\frac{\lambda_u}{2}\left\|\mathbf{u}_i\right\|_2^2+\sum_{j=1}^{N}\frac{\lambda_v}{2}\left\|\mathbf{v}_j\right\|_2^2+\frac{\lambda_{\mathbf{p}}}{2}\|\mathbf{p}\|_2^2+\frac{\lambda_{\mathbf{q}}}{2}\|\mathbf{q}\|_2^2. \end{aligned} \tag{4}$$

In order to update the embeddings of users and items efficiently in the online training module, we optimize Eq. (4) by the Alternating Least Square (ALS) technique [64] to obtain closed-form solutions of u_i and v_j, which are given as follows:

$$\begin{aligned} \mathbf{u}_i &= \left(\mathbf{V}_{\mathrm{p}}{}^{\top}\mathbf{V}_{\mathrm{p}}+\gamma\mathbf{V}_{\mathrm{q}}{}^{\top}\mathbf{V}_{\mathrm{q}}+\lambda_u\mathbf{I}\right)^{-1}\left(\mathbf{V}_{\mathrm{p}}{}^{\top}\mathbf{R}_i+\gamma\mathbf{V}_{\mathrm{q}}{}^{\top}\mathbf{Y}_i\right), \\ \mathbf{v}_j &= \left(\mathbf{U}_{\mathrm{p}}{}^{\top}\mathbf{U}_{\mathrm{p}}+\gamma\mathbf{U}_{\mathrm{q}}{}^{\top}\mathbf{U}_{\mathrm{q}}+\lambda_v\mathbf{I}\right)^{-1}\left(\mathbf{U}_{\mathrm{p}}{}^{\top}\mathbf{R}_{\cdot j}+\gamma\mathbf{U}_{\mathrm{q}}{}^{\top}\mathbf{Y}_{\cdot j}\right), \end{aligned} \tag{5}$$

where

$$\begin{aligned} \mathbf{V}_{\mathrm{p}} &= \mathbf{V}\,\mathrm{diag}(\mathbf{p}), \\ \mathbf{V}_{\mathrm{q}} &= \mathbf{V}\,\mathrm{diag}(\mathbf{q}), \\ \mathbf{U}_{\mathrm{p}} &= \mathbf{U}\,\mathrm{diag}(\mathbf{p}), \\ \mathbf{U}_{\mathrm{q}} &= \mathbf{U}\,\mathrm{diag}(\mathbf{q}). \end{aligned}$$

ALS iteratively optimizes one of $\mathbf{U}$ and $\mathbf{V}$ while the other one is kept unchanged. Thus, in each turn of the LQS subsession and the GQS subsession, after $\mathbf{Y}_i$ is updated according to u_i's feedback, we update $\mathbf{u}_i$ and $\mathbf{v}_j$ with Eq. (5). Note that this update process only affects the current conversational session, and not any follow-up user interactions.

4.3 The Local Question Search Subsession

In the LQS subsession, we construct questions for users from the exploitative strategy, i.e., by leveraging attributes of the items that users have interacted with. To achieve this goal, we first use the user-item interactions and item-attribute association to construct a directed graph. Formally, we denote this directed graph as $\mathcal{G}=(\mathcal{N},\mathcal{E})$, with a node-set $\mathcal{N}=\{\mathcal{U},\mathcal{V},\mathcal{A}\}$ and an edge set $\mathcal{E}$ constructed with rules. Note that, there are two types of relations associated with the corresponding edges in this directed graph:

- Interaction relations: If $u_i\in\mathcal{U}$ and $v_j\in\mathcal{V}$, $e_{ij}\in\mathcal{E}$ means u_i has rated v_j;
- Inclusive relations: if $v_j\in\mathcal{V}$, $a^l\in\mathcal{A}$, $e_{jl}\in\mathcal{E}$ means item v_j contains attribute a^l;

In this directed graph, there could be more than one path from a user node to a particular attribute node. For example, as shown in Figure 4, there are three paths from u_1 to a_2 (i.e., $u_1\rightarrow v_2\rightarrow a_2$, $u_1\rightarrow v_3\rightarrow a_2$ and $u_1\rightarrow v_4\rightarrow a_2$). We denote the number of paths from user i to attribute k as d_{ik} and assume that the more paths from a user to a particular attribute, the more the attribute can

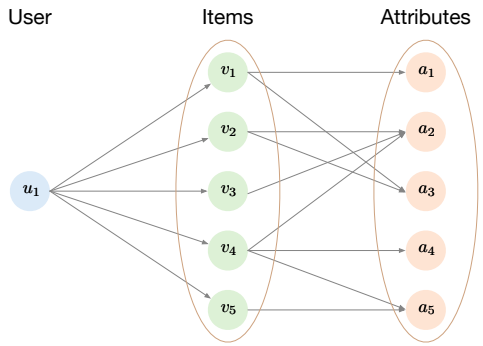


Fig. 4. Illustration of multiple paths from a user to certain attributes.

reflect the user's long-term preference. Thus, given a user u_i, we sort all the attributes from largest to smallest according to d_{ik}, and denote this ranked list of attributes as a question set L_{u_i}.

In the LQS subsession, we first initialize $\mathbf{u}_i$, $\mathbf{v}_j$ and $\mathbf{p}$ with the offline learned parameters as mentioned earlier. In the l-th turn, we select the l-th attribute from the question set L_{u_i} as the question for user u_i in this turn and wait for the response. After receiving u_i's feedback, we first update $\mathbf{Y}$, then update the $\mathbf{u}_i$ and $\mathbf{v}_j$ with Eq. (5). After the iterations in LQS subsession are done, we obtain the representations of $\mathbf{U}$ and $\mathbf{V}$ from the LQS subsession and denote them as $\mathbf{U}^L$ and $\mathbf{V}^L$. The attributes searched to construct questions in this subsession only interact with specific users, which means these attributes are part of the whole attribute pool and stand for the user's preferences, so we call it the LQS method.

4.4 The Global Question Search Subsession

In the GQS subsession, our RFCR model constructs questions from the exploratory strategy, i.e., by leveraging all the attributes in the dataset. To achieve this goal, we first complement the rating record of the user who is currently in the conversation session based on both the user's historic ratings and all historic ratings of items. Then we utilize the complemented record to construct the initial belief of user u_i's preference for items. In each turn of conversation, we update this belief according to the feedback of the user and then utilize it to select the attribute for the next turn.

4.4.1 Complement Rating Record. We utilize $\mathbf{R}$ to fill the zero-elements in vector $\mathbf{R}_i$, and denote the complemented vector as $\mathbf{R}_i^*$. When filling the zero-element in $\mathbf{R}_{ij}$, we consider two aspects: (1) the rating records of u_i; (2) the rating records of v_j. The intuition behind this is to consider both the propensity in the user's ratings and the popularity of the item.

Specifically, we let μ_{u_i} and σ_{u_i} represent the mean and variance of u_i's the rating records, and μ_{v_j} and σ_{v_j} represent the mean and variance of rating records of v_j. Then we utilize the normal distribution to sample $r^*_{u_i}$ and $r^*_{v_j}$ with the above parameters. Finally, we combine $r^*_{u_i}$ with $r^*_{v_j}$ with weights to produce $\mathbf{R}^*_{ij}$:

$$\mathbf{R}^*_{ij} = \mathbb{1}(\mathbf{R_{ij}} = 0)(\beta * r^*_{u_i} + (1-\beta) * r^*_{v_j}) + \mathbb{1}(\mathbf{R}_{ij} \neq 0)\mathbf{R}_{ij},$$
$$\beta = \frac{c_{u_i}}{c_{u_i} + c_{v_j}},$$
$$r^*_{u_i} = \mathcal{N}(r_{u_i}|\mu_{u_i}, \sigma_{u_i}),$$
$$r^*_{v_j} = \mathcal{N}(r_{v_j}|\mu_{v_j}, \sigma_{v_j}),$$

where c_{u_i} is the ratio of items rated by user u_i, and c_{v_j} is the ratio of users who rated item j:

$$c_{u_i} = \frac{\sum_{j=1}^{N} \mathbb{1}(\mathbf{R}_{ij} \neq 0)}{N},$$
$$c_{v_j} = \frac{\sum_{k=1}^{M} \mathbb{1}(\mathbf{R}_{kj} \neq 0)}{M}.$$

The intuition of introducing weights is that we believe users/items with more rating records would have higher confidence in their rating records.

4.4.2 Select Questions. We model the user preference for items in the GQS subsession by a (multinomial) probability distribution π^* over items $\mathcal{V}$. We also assume that there is a belief over the user preference $\boldsymbol{\pi}^*$ in each turn of the GQS subsession, which is a probability density function over all possible realizations of $\boldsymbol{\pi}^*$. We define the belief over $\boldsymbol{\pi}^*$ prior to the l-th turn in the GQS subsession as:

$$\mathbf{P}_l = Dir(\boldsymbol{\alpha} + \mathbf{Y}_i^{l-1}), \tag{6}$$

where $\mathbf{P}_l$ is a Dirichlet distribution with parameter $\boldsymbol{\alpha} + \mathbf{Y}_i^{l-1}$. Having applied the rating record completion, items can be ranked for user u_i based on $\mathbf{R}_i^*$, where the rank of each item can express our initial belief on the preference of items for each given user. Thus, we utilize the ranking information to initialize the hyperparameter $\boldsymbol{\alpha}$ of the Dirichlet distribution. In particular, we set α_j for item j to $\frac{1}{p_j+1}$, where p_j is the index of item j in the ranked list.

Finally, we can compute the user preference $\pi_l^*(v_j)$ over item v_j prior to the l-th question by:

$$\pi_l^*(v_j) = \mathbb{E}_{\pi \sim P_l}[\pi(v_j)] \quad \forall v_j \in \mathcal{V}. \tag{7}$$

Since $\boldsymbol{\pi}^*$ is a multinomial distribution over items $\mathcal{V}$, and $\mathbf{P}_l$ is modeled by the conjugate prior of the multinomial distribution (i.e. the Dirichlet distribution), from the properties of the Dirichlet distribution, the user preference $\boldsymbol{\pi}_l^*$ can be updated by counting and re-normalization of $\boldsymbol{\alpha}$ and $\mathbf{Y}_i^{l-1}$:

$$\pi_l^*(v_j) = \frac{\alpha_j + Y_{ij}^{l-1}}{\sum \alpha_j' + Y_{ij'}^{l-1}} \quad \forall v_j \in \mathcal{V}. \tag{8}$$

After asking the user questions and incorporating his/her response, the predicted belief and preference of the user are updated accordingly to his/her feedback. This belief tracker specifies the direction for moving towards the true underlying belief distribution and true user preference. This predicted user preference will guide the direction of question selection.

Following [72], we apply the greedy generalized binary search method (GBS) [41] to find the attribute that best splits the preference distribution $\boldsymbol{\pi}_l^*$ closest to two halves for the current candidate items during the l-th question

$$a^l = \arg\min_a \left| \sum_{v_j \in C_l} \left(2\mathbb{1}\left\{a \in \mathcal{A}_v\right\} - 1\right) \boldsymbol{\pi}_l^*(v) \right|, \tag{9}$$

where a^l is the chosen attribute in the l-th turn. C_l is the candidate items set in the l-th turn, which is initialized as $\mathcal{V}$ ($C_1 = \mathcal{V}$). The update rule for C_l is

$$C_{l+1} = C_l \cap \{v_j | a_j^l = a_*^l\}.$$

Similar to the LQS subsession, we initialize the $\mathbf{u}_i$, $\mathbf{v}_j$, and $\mathbf{p}$ with the offline learned parameters at the start of the GQS subsession. In the l-th turn, we compute the preference distribution with Eq. (8), then select the l-th attribute with the GBS method as the question to ask u_i and wait for the user's response. After receiving u_i's feedback, we first update $\mathbf{Y}$, then update the $\mathbf{u}_i$ and $\mathbf{v}_j$

Algorithm 1: The proposed algorithm in RFCR.

Input: Q_1, Q_2; $\mathbf{R}$; Directed graph $\mathcal{G}$; Offline learned $\mathbf{U}$ and $\mathbf{V}$; u_i
Output: Recommendation list;
`// Prepare for the Local Question Search subsession`
Compute $\{d_{ik}\}_{k=1}^{L}$ in $\mathcal{G}$;
Get L_{u_i} via sorting $\{d_{i,k}\}_{k=1}^{L}$;
$\mathbf{U}^L \leftarrow \mathbf{U}, \mathbf{V}^L \leftarrow \mathbf{V}, \mathbf{Y} \leftarrow \mathbf{0}$;
`// The Local Question Search subsession starts`
for *turn* $l = 1, 2, \ldots, Q_1$ **do**
    Select the l-th attribute a^l in L_{u_i};
    Ask the question about a_l, observe the reply a_*^l;
    Remove a^l from question pool;
    Update $\mathbf{Y}_i$ by the reply a_*^l according to Eq. (1) and Eq. (2);
    Update $\mathbf{U}^L$ and $\mathbf{V}^L$ by ALS according to Eq. (5);
end
`// Prepare for the Global Question Search subsession`
Get R_i^* via complement rating record;
Get $Ranking_l$ via ranking items according to R_i^*;
$\boldsymbol{\alpha} \leftarrow Ranking_l$;
$C_1 \leftarrow \mathcal{V}$;
$\mathbf{U}^G \leftarrow \mathbf{U}, \mathbf{V}^G \leftarrow \mathbf{V}, \mathbf{Y} \leftarrow \mathbf{0}$;
`// The Global Question Search subsession starts`
for *turn* $l = 1, 2, \ldots, Q_2$ **do**
    Compute the user belief $\mathbf{P}_l$ via Eq. (6);
    Compute the user preferences $\boldsymbol{\pi}_l^*$ w.r.t Eq. (9);
    Select the l-th attribute a_l via Eq. (7);
    Ask the question about a_l, observe the reply a_*^l;
    Remove a^l from question pool;
    $C_{l+1} = C_l \cap \{v_j | a_j^l = a_*^l\}$;
    Update $\mathbf{Y}_i$ by the reply a_*^l according to Eq. (1) and Eq. (2);
    Update $\mathbf{U}^G$ and $\mathbf{V}^G$ by ALS according to Eq. (5);
end
Fuse the embeddings in two subsessions w.r.t Eq. (10);
Generate recommendation list by fused embeddings with inner product;

with Eq. (5). After the iterations in GQS subsession are done, we obtain the representations of $\mathbf{U}$ and $\mathbf{V}$ from the GQS subsession and denote them as $\mathbf{U}^G$ and $\mathbf{V}^G$. The attributes used to construct questions in this subsession come from all attributes in the dataset, so we call it the GQS method.

4.5 Preference Fusion and Recommendation

After the above two subsessions, we can obtain the updated embeddings of users and items in both the LQS subsession and the GQS subsession. Then, we fuse the user and item embeddings from these two subsessions with linear concatenation respectively, which are given by:

$$\begin{aligned} \mathbf{U} &= \lambda \mathbf{U}^L + (1-\lambda)\mathbf{U}^G, \\ \mathbf{V} &= \lambda \mathbf{V}^L + (1-\lambda)\mathbf{V}^G, \end{aligned} \tag{10}$$

Table 2. Statistics of the three evaluated datasets.

Dataset	#users	#items	#attributes	#ratings	density
Movielens	610	9,742	1,601	100,836	1.70%
BookCrossing	902	6,929	2,900	102,263	1.64%
Anime	1,367	4,916	4,696	114,618	1.71%

where λ is a hyperparameter calculated by:

$$\lambda = \frac{Q_1}{Q_1 + Q_2}.$$

Finally, the recommendation list is generated by sorting the inner product of the user embedding $\mathbf{u}_i$ and item embeddings in $\mathbf{V}$.

Algorithm 1 shows the whole process of RFCR in a conversation session. Before the LQS subsession, we get the ranking of attributes (lines 1-2). Then in each turn of the LQS subsession, we ask the user with the corresponding attribute retrieved by the directed graph and get feedback (lines 5-6), and finally update the affinity matrix and embeddings (lines 8-9). Next, before the GQS subsession, we initialize the belief and candidate set (lines 11-14). Then in each turn of the subsession, we first compute the user preference distribution and select attributes based on it (lines 17-20), then we get the reply from the user and update the candidate set, affinity matrix, and embeddings (lines 20-24). After these two subsessions, the final representations are fused by linear concatenation and are leveraged by the model to generate the recommendation list for the user (lines 26-27).

5 EXPERIMENTS

In this section, we first state our research questions in § 5.1. We then introduce the datasets in § 5.2 and describe the evaluation metrics in § 5.3. In § 5.4, we describe the implementation details of our RCFR. In § 5.5, we introduce the user simulator employed in this paper. Finally, we introduce the baselines in § 5.6.

5.1 Research Questions

We use the following research questions (RQs) to guide the remainder of the paper:

RQ1 What is the impact of the splitting schemes of Q on the performance of our RFCR model?
RQ2 How does our RFCR model perform compared with existing conversational recommendation methods?
RQ3 What is the impact of the total number of turns in conversation on the performance of RFCR?
RQ4 What are the effects of key components in CRS, i.e., LQS, GQS?
RQ5 How does our RFCR perform on the cold-start problem?
RQ6 How interpretable are the questions for users asked by our RFCR model?

5.2 Datasets

We conduct experiments on three publicly available datasets from three different domains, namely Movielens-latest [14] [4] for movie recommendation, BookCrossing [4] [5] for book recommendation and Anime [48] [6] for anime recommendation. Details of the datasets are as follows:

- **Movielens-latest**: MovieLens datasets were collected by the GroupLens Research at the University of Minnesota from the MovieLens website. MovieLens allows its users to submit

[4]http://movielens.org/
[5]https://grouplens.org/datasets/book-crossing/
[6]https://www.kaggle.com/azathoth42/myanimelist

ratings and reviews for movies they have watched and recommend movies that users may enjoy. The Movielens-latest dataset originally includes 27,753,444 ratings and 1,108,997 tag applications across 58,098 movies. These data were created by 283,228 users between January 09, 1995, and September 26, 2018. This dataset was generated on September 26, 2018. All selected users had rated at least 1 movie. Each movie in this dataset is associated with basic information (*title* and *genres*). The rating score ranges from 0.5 to 5 with 0.5 granularity. For brevity, we denote this dataset as **Movielens**.

- **BookCrossing**: The BookCrossing dataset was collected by Cai-Nicolas Ziegler from the Book-Crossing community. It contains 1,149,780 ratings from 278,858 users for 271,379 books. The ratings are either explicit, denoted by a scale from 1-10 (higher value denotes higher appreciation), or implicit, denoted by 0. All selected users had rated at least 1 book. Each book in this dataset is associated with basic information (*Location, ISBN, Book-Title, Book-Author, Year-Of-Publication* and *Publisher*). The rating score ranges from 1 to 10 with 1 granularity.
- **Anime**: The Anime dataset was collected by Matěj Račinský [7] from the Myanimelist [8]. It contains 7,813,737 ratings from 73,516 users for 12,292 animes. The ratings are either explicit, expressed on a scale from 1-10 (higher value denotes higher appreciation), or implicit, denoted by -1 if the user watched it but did not assign a rating. All selected users had rated at least 1 anime. Each anime in this dataset is associated with basic information (*ID, Name, Author, Genre, Type, Episodes, Ratings* and *Members*). The rating score ranges from 1 to 10 with 1 granularity.

In GQS subsession, we propose to complement the user-item interaction records and construct the initial belief π^* of the user's preference for items. To approximate the initial belief on the preference of items as closely as possible, we need to get more original rating records of user u_i and rating records for item v_j. Furthermore, considering the huge size and extreme sparsity of the original dataset, we filter the dataset in the same way mentioned in [16] to construct the user-item recommendation matrix, where we retain users with at least 200 interactions and items with at least 100 interactions.

For all three datasets, following [49], we first filter out attributes that are not able to be utilized for constructing questions (e.g., non-textual data like ISBN). In addition,

- In **Movielens**, the tags that were assigned to products by users objectively represent the preferences of users and the characteristics of films. Thus, we take tag applications and *genres* information as the attributes of movies. We filter out the original data with the rules above and ratings of 0. After the filtering, there are 610 users, 9,742 items, and 100,836 rating records left in the dense subset.
- In **BookCrossing**, similarly, we use *Book-Author, Publisher* and *Year-Of-Publication* as the attributes of books. We filter out original data with the rules above and ratings of 0. After the filtering, there are 902 users, 6,929 items, and 102,263 rating records left in the dense subset. To avoid a slow or unstable learning process, we scale all the rating scores into the range of $[0, 5]$.
- In **Anime**, as the same as the operation process in BookCrossing, we select *Genre, Author* and *Type* as the attributes of animes. We filter out the original data with the rules above and ratings of -1. After the filtering, there are 1,367 users, 4,916 items, and 114,618 rating records left in the dense subset. To avoid a slow or unstable learning process, we scale all the rating scores into the range of $[0, 5]$.

[7] https://github.com/racinmat
[8] https://myanimelist.net/

The statistics of the datasets are shown in Table 2. Finally, we randomly split the user-item ratings into training, validation, and testing sets with a ratio of 7:2:1 for all of the three datasets [49, 61]. We report the average evaluation results over 10 runs.

5.3 Evaluation Metrics

To evaluate the performance of our RFCR model, we employ two different types of evaluation metrics, i.e., ranking-oriented metrics and conversation-oriented metrics, for evaluating the quality of recommendation lists generated from CRSs and the quality of conversational sessions in CRSs.

- **Ranking-oriented Evaluation Metrics.** For evaluating the performance of recommendation lists, we let $\mathcal{I}_{\text{test}}$ be the user-item pairs in the test set. In the test stage, for each user-item pair (u, v) in $\mathcal{I}_{\text{test}}$, we set v as the target item, and then ask user u some questions to obtain u's feedback as we described in § 3.2. Finally, we generate a recommendation list to the user u who targets at item v, and denote it as $l_{(u,v)}$. For each user in a test dataset, we compute evaluation metrics based on the recommendation lists generated by each model. The ranking-oriented evaluation metrics we used are recall at cut-off k ($Recall@k$), mean average precision at cut-off k ($MAP@k$), mean reciprocal rank (MRR) and normalized discounted cumulative gain at cut-off k ($NDCG@k$) [22]. Details of how to obtain the evaluation scores using these ranking-oriented evaluation metrics are shown as follows.
  - **Recall@k** is the proportion of relevant items found within the top-k recommended items. For each recommendation list $l_{(u,v)}$, the calculation of the $Recall_{(u,v)}@k$ is:

$$RI_{(u,v)}@k = \sum_{i=1}^{k} \mathbb{1}\left\{l_{(u,v)}(i) \in \mathcal{V}_{u,\text{test}}\right\},$$

$$Recall_{(u,v)}@k = \frac{RI_{(u,v)}@k}{\left|\mathcal{V}_{u,\text{test}}\right|},$$

    where $l_{(u,v)}(i)$ is the item at position i in $l_{(u,v)}$, and $\mathcal{V}_{u,\text{test}}$ is the set of items that user u has rated in the test set. The average Recall@k among the test set is:

$$Recall@k = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{(u,v)\in\mathcal{I}_{\text{test}}} Recall_{(u,v)}@k,$$

  - **MAP@k** is the mean $AP@k$ value among the test set, and $AP@k$ is defined as the average precision computed at each position of relative items in the top k items of the user's ranked recommendation list. Compared with recall and precision, average precision considers the order in which the returned items are presented. For each recommendation list $l_{(u,v)}$, the calculation of the $AP_{(u,v)}@k$ is:

$$AP_{(u,v)}@k = \frac{1}{N_u(k)} \sum_{i=1}^{k} \mathbb{1}\left\{l_{(u,v)}(i) \in \mathcal{V}_{u,\text{test}}\right\} P_{(u,v)}@i,$$

    where

$$P_{(u,v)}@k = \frac{RI_{(u,v)}@k}{k},$$

$$N_u(k) = \min\left(k, \left|\mathcal{V}_{u,\text{test}}\right|\right).$$

    The $MAP@k$ among the test set is:

$$MAP@k = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{(u,v)\in\mathcal{I}_{\text{test}}} AP_{(u,v)}@k.$$

– **MRR** is the average of reciprocal ranks of the desired items. The reciprocal rank is set to zero if the rank is larger than k. *MRR* takes the rank of the item into account, which is important in settings where the order of recommendations matters. For the whole test set, the calculation of the *MRR* is:

$$MRR = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{(u,v) \in \mathcal{I}_{\text{test}}} \frac{1}{\text{rank}_{(u,v)}},$$

where $\text{rank}_{(u,v)}$ refers to the rank position of the *first* relevant item in $l_{(u,v)}$.

– **NDCG@k** is a position-aware metric that assigns more weights to the correct items that rank higher in a recommendation list [22], and it penalizes the effectiveness score in a smoother way compared to *MRR*. For each recommendation list $l_{(u,v)}$, the calculation of the $NDCG_{(u,v)}@k$ is:

$$DCG_{(u,v)}@k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)},$$
$$IDCG_{(u,v)}@k = \sum_{i=1}^{|REL_k|} \frac{rel_i}{\log_2(i+1)},$$
$$NDCG_{(u,v)}@k = \frac{DCG_{(u,v)}@k}{IDCG_{(u,v)}@k},$$

where $rel_{(u,v)}(i)$ is the graded relevance score of the item at position i in $l_{(u,v)}$. In our experiment, we set $rel_{(u,v)}(i)$ to 1 if the $l_{(u,v)}(i) \in \mathcal{V}_{u,\text{test}}$, and otherwise 0. Similarly, $irel_{(u,v)}(i)$ is the graded relevance score of the item at position i in $il_{(u,v)}$, where $il_{(u,v)}$ represents the ideal recommendation list (ordered by graded relevance score of items in $l_{(u,v)}$) in the set of items. Thus, a $NDCG_{(u,v)}@k$ score is always in the range of [0,1], with higher values indicating better retrieval performance. And the average $NDCG@k$ among the test set is:

$$NDCG@k = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{(u,v) \in \mathcal{I}_{\text{test}}} NDCG_{(u,v)}@k,$$

Therefore, a higher $Recall@k$, $MAP@k$, *MRR*, and $NDCG@k$ indicate a higher performance when these models produce the final recommendation lists. In our experiments, we set the k to 10. Note that using the top 10 items to calculate *MAP* and *Recall* means that it is convenient to avoid baselines from nearly zero *MAP* and *Recall* scores.

- **Conversation-oriented Evaluation Metrics.** For evaluating the performance of conversational sessions, we use the success rate ($SR@t$) [55] to measure the ratio of successful conversations, i.e., recommend the ground truth item by turning t. From the interactive dialogue system's perspective, the successful rate as a conversation-oriented metric can be viewed as the conversion rate, which can be formulated as:

$$SR@t = \frac{\#\ successful\ conversations\ by\ turn\ t}{\#\ conversations}.$$

We also report the average turns (AT), i.e., the average of minimum turns that are needed to terminate the session. Larger SR denotes better recommendation and smaller AT denotes more efficient conversation.

5.4 Implementation Details

We implement our RFCR model by Pytorch. [9] We set the hyper-parameters for offline training and online updating phases, respectively.

For offline training, we set the dimension of user and item embeddings D to 256. We use the Adam optimizer [26] as the optimizer of our model, whose learning rate is searched in this parameter set $\{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$. As the offline training module does not leverage the user feedback information matrix Y, we set γ and $\lambda_{\mathbf{q}}$ to 0. We set the rest hyper-parameters as follows: the training iterations in offline training are tuned from 20 to 200 with a step size of 20 ; λ_u, λ_v, and $\lambda_{\mathbf{p}}$ selected from 0.05 to 0.4 with step size 0.05 . After performing hyper-parameter grid search, we find the optimal hyper-parameters as follows: the learning rate is 10^{-3}, the training iterations in offline training is 100 and $\lambda_u = \lambda_v = \lambda_{\mathbf{p}} = 0.1$. The above hyper-parameter searching process is also performed in the online training stage. For online training, following the settings in 3, we use the initial user and item embeddings learned from offline training and set the rest hyper-parameters as follows: γ is set to 1 and $\lambda_u = \lambda_v = \lambda_{\mathbf{p}} = \lambda_{\mathbf{q}} = 0.1$. We will discuss the splitting schemes of total turns Q in § 6.1.

5.5 User Simulator For CRS

For evaluation purposes, we follow the settings of previous works, which apply a user simulator to simulate one conversation session for each user in a CRS [55, 67]. By doing so, we are able to collect interaction data for both the training and evaluation stages. Following the previous works [30, 73], two rules are applied in an interactive experiment: 1) A user will only accept the items which influence his/her preferences; 2) A user will answer "yes" to one question if this question is constructed by the attributes that are from his/her favorite items. One major attack for such simulation is that the user may "falsely" reject an item that is liked by him/her but has not been observed hence not being interacted with him/her. It is not appropriate to employ such simulation in a CRS, but this is the most practical and realistic at the current stage.

5.6 Baselines and Settings

To evaluate the performance of our proposed RFCR model, we compare our RFCR with two different categories of baseline models, which are listed in the following.

- General recommendation models:
  - Neural Collaborative Filtering (NCF) [16]: This method is a state-of-the-art neural collaborative filtering (CF) model, which uses multiple hidden layers upon the element-wise and concatenation of user and item embeddings to capture their nonlinear feature interactions.
  - Neural Graph Collaborative Filtering (NGCF) [57]: This method adopts three graph neural network (GNN) layers on the user-item interaction graph, aiming to refine user and item representations via at most three-hop neighbors' information.
  - LightGCN [17]: This method is a graph convolutional neural network based static CF algorithm which simplifies but outperforms the NGCF method by removing feature transformation and nonlinear activation.
- Conversational recommendation models:
  - **Max Entropy (ME)**: This method follows a rule-based protocol to select attributes. In each turn, a CRS retrieves the attribute that has the maximum entropy among the candidate items. We emphasize that we use the same representation offline training and online updating method as RFCR in this baseline.

[9] https://pytorch.org/.

Table 3. Time complexity comparison of conversational recommendation methods, where, as defined in the above, N is the number of items, M is the number of users, L is the number of attributes, I is the total number of user-item interactions (i.e., $I = \sum_{i=1}^{M} \sum_{j=1}^{N} \mathbb{1}(\mathrm{R}_{ij} > 0)$), C is the total number of item-attribute associations (i.e., $C = \sum_{i=1}^{N} |\mathcal{A}_i|$), L_g is the number of layers in Graph Neural Network (GNN), L_p is the number of layers in Policy Network, D_p is the dimension of hidden layers in Policy Network, D is the dimension of embeddings and K is the number of negative samples.

Stage	EAR	SCPR	FPAN	Max Entropy	Qrec	RFCR
Embedding Training (each iteration)	$O(K(I+C)D))$	$O(K(I+C)D))$	$O(L_g((M+N+L)D^2+(I+C)D))$	$O(ID)$	$O(ID)$	$O(ID)$
Policy Network Training (each iteration)	$O(L_pD_p^2)$	$O(L_pD_p^2)$	$O(L_pD_p^2)$	–	–	–
Online Embedding Updating (each turn)	$O(D)$	$O(D)$	$O(D^3)$	$O((M+N+D)D^2)$	$O((M+N+D)D^2)$	$O((M+N+D)D^2)$
Online Attribute Selection (each turn)	$O(LD)$	$O(L_pD_p^2)$	$O(L_pD_p^2)$	$O(NL)$	$O(NL)$	$O(NL)$ or $O(1)$

Table 4. Average training time comparison over different CRS methods on three datasets. The best performance per column is in boldface.

	Movielens	BookCrossing	Anime
Max Entropy	35min44s	49min59s	55min06s
EAR	40min18s	66min58s	81min06s
Qrec	41min25s	42min26s	55min17s
SCPR	80min03s	63min05s	66min56s
FPAN	40min10s	54min49s	90min49s
RFCR	**23min02s**	**24min51s**	**29min54s**

- **EAR** [30]: This model is proposed with a three-stage solution called Estimation-Action-Reflection. It trains a policy network by reinforcement learning to decide whether to ask questions or recommend items. It uses a pretrained offline factorization machine (FM) model to decide which item to recommend and employs a policy network trained by policy gradient to decide the conversational policy.
- **Qrec** [74]: This method infers the underlying user belief and preference over items to learn the question-asking strategy and asks users a sequence of questions based on all attributes.
- **SCPR** [31]: This method models conversational recommendation as an interactive path reasoning problem on a graph. It walks through the attribute vertices by following user feedback, explicitly utilizing the user's preferred attributes.
- **FPAN** [61]: This model makes use of Graph Neural Networks (GNN) to learn the offline user-item initial representations and designs two gating modules to aggregate the online item-level feedback and attribute-level feedback information considering the relations between feedback signals.

In order to demonstrate whether our RFCR model could reduce the time complexities during training compared to other CRS methods with policy agents, we summarize the time complexities of the EAR, Qrec, SCPR, FPAN and RFCR in Table 3. As shown in the table, we can find that our RFCR model has the lowest time consumption during training. In addition, in order to verify the correctness of our summary, we also collect the real runtime of these CRS methods when they are deployed in our experiments. All experiments are run on GeForce RTX 2080 Ti GPU, and the results are shown in Table 4.

Additionally, in order to verify the key designs of our RFCR model, we take into account the following variants of our proposed RFCR model for comparisons:

- **RFCR-R-L**, where we only randomly select attributes to construct questions for users in the LQS subsession.

- **RFCR-R-G**, where we only randomly select attributes to construct questions for users in the GQS subsession.
- **RFCR-W-L**, where we remove the LQS subsession and only keep the GQS subsession in the whole conversation session (i.e., $Q_2 = Q$).
- **RFCR-W-G**, where we remove the GQS subsession and only keep the LQS subsession in the whole conversation session (i.e., $Q_1 = Q$).

For the Max-Entropy and Qrec, we use the same parameter settings as our RFCR model. For the EAR, SCPR and FPAN, we use the optimal parameters reported in the corresponding paper and tune their hyper-parameters in the same way as reported. In addition, we set the dimension of representations D of all baseline models and variants of RFCR to 256 for fair comparisons. Although there are other recent conversational recommendation methods [7, 8, 35, 36, 46, 47, 49, 65], they are ill-suited for comparison because of their different task settings. In addition, for fair comparisons, the baselines also obey the same settings we set in § 3, where a CRS only asks users questions multiple times and recommends only once at the end of the conversation session.

6 RESULTS AND ANALYSIS

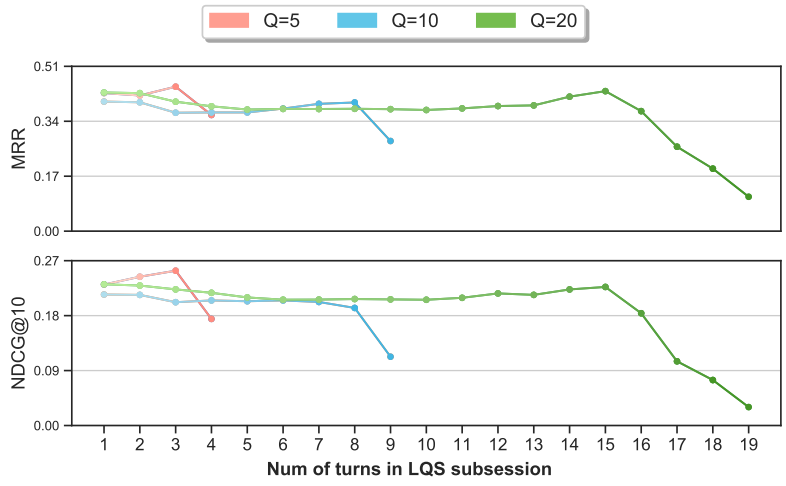
In this section, we provide the experimental results and the analysis in response to the research questions listed in § 5.1.

6.1 Impact of Different Splitting Schemes of Q (RQ 1)

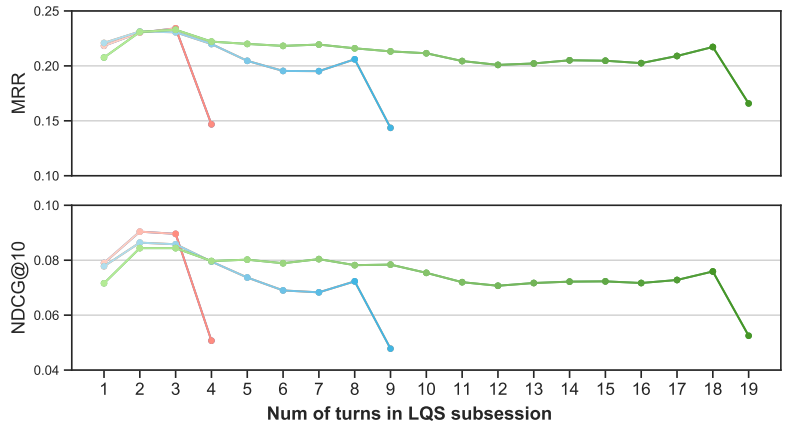
In our proposed RFCR model, the questions asked to users (the number of which is Q) in the conversational session comprise those based on local attributes (the number of which is Q_1) and those based on global attributes (the number of which is Q_2), where $Q = Q_1 + Q_2$. Since the datasets in our experiments have different features, e.g., the total number of attributes in Movielens are smaller than those in Anime, it is not reasonable to set a uniform splitting scheme of Q for all the datasets. In order to find the optimal splitting scheme for each dataset, we first design an experiment to study the impact of different splitting schemes of Q for each dataset. Considering the limited number of users' interactions in CRS, we set Q to $\{5, 10, 20\}$, respectively. There are several schemes to split Q into Q_1 and Q_2, we choose the greedy search method (e.g., when $Q = 5$, (Q_1, Q_2) could be $(1, 4)$, $(2, 3)$, $(3, 2)$ or $(4, 1)$) to find the optimal splitting scheme. Note that we set neither Q_1 nor Q_2 to 0 in this section, where 0 denotes that either the LQS subsession or the GQS subsession is removed, which we will discuss later in § 6.4. We use **NDCG@10** and **MRR** as our representative evaluation metrics in this experiment, as the same trends can be observed when evaluating the performance on other evaluation metrics.

The experimental results are provided in Figure 5. We have the following findings:

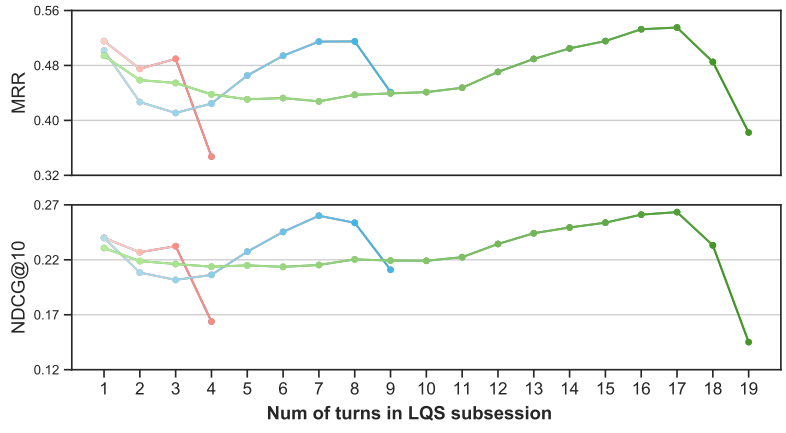
- (1) In terms of the Movielens dataset, when increasing Q_1, the performance of our RFCR model changes smoothly, and then it achieves a plato when Q_1 reaches ~75% Q (e.g., $Q = 20$ and $Q_1 = 15$). Subsequently, the performance drops dramatically when $Q_1 \geq 75\%Q$ (e.g., $Q = 20$ and $Q_1 \geq 15$). This is because when less than $25\%Q$ questions are asked in the GQS subsession (i.e., $Q_1 \geq 75\%Q$), the user preference $\boldsymbol{\pi}^*$ is determined by $\boldsymbol{\alpha}$ and few users' feedback. Thus, we cannot construct proper questions for users in the GQS subsession and get a high-quality representation of users' preferences. Accordingly, we set $(Q_1, Q_2) = (75\%Q, 25\%Q)$ to be the optimal splitting scheme in the Movielens dataset.
- (2) In terms of the BookCrossing dataset, our RFCR model achieves the best performance when Q_1 is 3 regardless of Q (e.g. $Q = 20$ and $Q_1 = 3$). When we ask users less than 2 global questions (i.e., $Q_1 \geq Q - 2$), the performance drops dramatically. It means that most users have 2 or 3 relevant local attributes that can better infer the users' intrinsic purchase intentions in



(a) Movielens



(b) BookCrossing



(c) Anime

Fig. 5. Recommendation performance evaluated by the representative metrics, *MRR* and *NDCG*@10, for datasets, Movielens, BookCrossing, Anime, from top to the bottom, again the different number of turns, Q_1, in local question search subsession for different Q, i.e., $Q = 5, 10, 20$, respectively.

Table 5. Performance comparison of the bandit-based RFCR models and the RFCR model on three datasets by *Recall*@10, *MAP*@10, *MRR* and *NDCG*@10, where the best performance per column is boldfaced.

	Movielens				BookCrossing				Anime			
Method	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10
RFCR-linUCB	.0270	.1067	.4241	.2266	.0155	.0303	.1926	.0731	.0480	.1127	.4859	.2304
RFCR-conTS	.0041	.0158	.1391	.0404	.0057	.0063	.0684	.0176	.0038	.0035	.0414	.0107
RFCR	**.0281**	**.1273**	**.4470**	**.2536**	**.0171**	**.0379**	**.2342**	**.0896**	**.0507**	**.1197**	**.5155**	**.2400**

this dataset. Finally, we set the $(Q_1, Q_2) = (3, Q-3)$ to be the approximate optimal splitting scheme in the BookCrossing dataset.

- (3) In terms of the Anime dataset, the performance of our RFCR model reaches a plato when $Q = 20$ and Q_1 is close to the 75% of the total turns (e.g., $Q = 20$ and $Q_1 = 15$). Subsequently, the performance drops dramatically when $Q_1 \geq 75\%Q$ (e.g., $Q = 20$ and $Q_1 \geq 15$). In addition, when $Q_1 = 1$, it also achieves excellent performance. The Anime dataset has a larger attribute space for constructing questions than the other two datasets. As a result, when $Q_1 << Q_2$, we construct questions for users to capture their potential needs from the GQS subsession. However, with the increase of Q_1, We gradually concentrate on inferring the users' purchasing intentions from the LQS subsession. Accordingly, we set $(Q_1, Q_2) = (75\%Q, 25\%Q)$ to be the optimal splitting scheme in the Anime dataset.

Once again, Figure 5 demonstrates that it is not reasonable to set a uniform splitting scheme of Q in all scenarios. In the following experiments, we set the optimal splitting scheme of Q on each dataset according to the above analysis.

Furthermore, in order to explore the effectiveness of applying the classical methods for EE balance in our framework , we attempt to deploy two different types of bandit-based methods, named linUCB [34] and ConTs [36], into our RCFR models, where the linUCB method is used to find the optimal splitting scheme of Q. In contrast, the ConTS is used to directly find the optimal attributes used for constructing questions for users.

- linUCB is a state-of-the-art contextual bandit algorithm, which selects arms based on the upper confidence bound of the estimated reward with given context vectors. In our RFCR model, we employ the linUCB method to balance the splitting scheme of the GQS subsession and the LQS subsession in order to achieve the EE trade-offs. Specifically, we set each splitting scheme as an arm, set the user embeddings as context information, and regard the NDCG of the recommendation list as the reward for updating policy.
- ConTS seamlessly unify attributes and items in the same arm space and achieve their EE trade-off automatically using the framework of Thompson Sampling. In our RFCR model, we set each attribute as an arm and use the ConTS method to select attributes in the testing stage. After selecting an arm in each turn, we calculate the reward based on if the attribute belongs to the target item. We denote such method that integrates ConTS into our RFCR model as RFCR-ConTS.

Then, we also use *Recall*@10, *MAP*@10, *NDCG*@10, and *MRR* as our representative evaluation metrics. For fair comparisons, we fix the number of questions in the conversation session to 5 and the experiment results are shown in Table 5. It can be observed that whether the bandit-based method is used for finding the splitting scheme of Q or selecting attributes directly, the performance of our RFCR model cannot be further improved.

6.2 Performance Comparison between Our RFCR and the Baselines (RQ 2)

To answer how effective our proposed method is compared to prior works, we compare our model with three widely used general recommendation baseline models, named NCF [16], NGCF [57], LightGCN [17], and five state-of-the-art conversational recommendation baseline models, namely,

Table 6. Performance comparison of the three general recommendation models and our RFCR model on three datasets by *Recall*@10, *MAP*@10, *MRR* and *NDCG*@10. The best performance per column is boldfaced.

	Movielens				BookCrossing				Anime			
Method	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10
NCF	.0093	.0177	.1380	.0533	.0078	.0073	.0853	.0232	.0105	.0244	.2066	.0668
NGCF	.0111	.0252	.1571	.0631	.0112	.0143	.0984	.0337	.0346	.0444	.2664	.1236
LightGCN	.0122	.0269	.1744	.0659	.0119	.0162	.0997	.0359	.0358	.0527	.2943	.1298
RFCR	**.0281**	**.1273**	**.4470**	**.2536**	**.0171**	**.0379**	**.2342**	**.0896**	**.0507**	**.1197**	**.5155**	**.2400**

Table 7. Performance comparison of all CRS methods on three datasets by *Recall*@10, *MAP*@10, *MRR* and *NDCG*@10, where the best performance is boldfaced.

	Movielens				BookCrossing				Anime			
Method	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10
Max Entropy	.0046	.0198	.1424	.0445	.0041	.0067	.0711	.0211	.0056	.0014	.0955	.0291
EAR	.0032	.0123	.1387	.0465	.0033	.0072	.0831	.0255	.0230	.0492	.2391	.1017
Qrec	.0087	.0368	.1680	.0625	.0056	.0187	.1053	.0191	.0060	.0032	.1101	.0334
SCPR	.0242	.0763	.1785	.0648	.0158	.0307	.1166	.0349	.0439	.0928	.4180	.1940
FPAN	.0214	.0769	.1803	.0768	.0094	.0182	.1413	.0441	.0460	.0892	.3782	.1820
RFCR	**.0281**	**.1273**	**.4470**	**.2536**	**.0171**	**.0379**	**.2342**	**.0896**	**.0507**	**.1197**	**.5155**	**.2400**

Max Entropy [23], EAR [30], Qrec [74], SCPR [31], and FPAN [61]. For general recommendation baseline models, we only run them under the evaluation metrics of recommendation. However, for conversational recommendation baseline models, we fix the number of questions in the conversation session to 5 for fair comparisons and run them under both the evaluation metrics of recommendation and conversational sessions.

The performance of recommendation are presented in Table 6 and Table 7. We can see that our RFCR model achieves significantly higher performance on all recommendation metrics than that of the state-of-the-art baselines, demonstrating the superior performance of our RFCR model. It can be observed that these three general recommendation baseline models can not outperform our RFCR model, which indicates that our RFCR model can improve the performance over static models as interacting with users continuously can capture their real needs and produce more accurate recommendation results. Meanwhile, compared with conversational recommendation baseline models, RFCR model achieves better performance than the Max Entropy model. This result suggests that our proposed RFCR model is more effective than the single rule-based model, which uses one specific question search rule from scratch to the end. In contrast, we construct questions for users from two strategies and update the question search policies in the GQS subsession to capture more precise users' needs. Moreover, our model is superior to the EAR model. The datasets in our experiments have a large attribute space, which requires more training data to train the Reinforcement Learning (RL) module of the EAR model. Without an accurate attribute estimation module, the EAR model cannot find proper questions for users. However, we guide the construction of questions using the statistics calculated from the dataset, which releases the burden of training policy agents. It is also obvious from that Table 7 that our RFCR model performs better than the SCPR model. Although the graph in the SCPR model provides constraints and helps eliminate irrelevant attributes for constructing questions in the conversational session, the SCPR model still has terrible performance in large attribute space. In contrast, we construct local questions based on attributes that users have interacted with and find an approximate optimal scheme to balance the trade-off between global questions and local questions, so we can better infer users' purchasing intentions in a large attribute space. Additionally, Our RFCR model outperforms the Qrec model. The Qrec model asks questions based on users' initial preference on items from historic data, which causes that a CRS produces recommendation lists that are full of items that users inherently like and neglect the users' potential purchasing intentions. Different from the Qrec model, we ask users questions from both the exploitative and the exploratory strategies and propose to fuse the user

Table 8. Performance of our RFCR and baselines on three datasets, where the best performance per dataset per metric is in boldface.

	Movielens		BookCrossing		Anime	
Method	**SR@10**	**AT**	**SR@10**	**AT**	**SR@10**	**AT**
Max Entropy	0.0281	-	0.0278	-	0.0308	-
EAR	0.0232	5.701	0.0134	6.127	0.0239	8.164
Qrec	0.0313	-	0.0342	-	0.0342	-
SCPR	0.0281	4.877	0.0175	5.023	0.0492	7.141
FPAN	0.0360	3.932	0.0311	4.632	**0.0598**	5.024
RFCR	**0.0488**	-	**0.0354**	-	0.0559	-

representations from these two strategies for accurately capturing users' preferences. Finally, the FPAN model achieves a lower performance than our RFCR model. The FPAN model initializes the user-item representations with GNN and updates the policy of constructing questions for users from item-level feedback and attribute-level feedback without employing policy agents. However, the FPAN model only selects attributes or items to construct questions from the historic log data, so users' potential interests are neglected. Our RFCR model is able to capture users' potential purchasing intentions from the GQS subsession and capture users' intrinsic interests from the LQS subsession and then fuses them to capture the users' real needs.

As for the performance of conversational sessions, Table 8 shows the experimental results. Notably, we do not report the results of AT about Max Entropy, Qrec, and RFCR model in Table 8, since these three models generate recommendation lists for users at the end of conversation sessions, whose settings are not proper for these three models are evaluated by AT. However, from the results of SR@10, we still can observe that our RFCR can achieve better performance than most of the state-of-the-art baselines.

In what follows, we will focus on the evaluation of models related to the quality of recommendation in § 6.3, § 6.4 and § 6.5, respectively.

6.3 Effect of Total Number of Turns in Conversation (RQ 3)

To demonstrate that our RFCR model could achieve better performance with fewer turns, we set $Q = \{5, 10, 15, 20\}$ respectively and make performance comparisons among MaxEntropy, EAR, Qrec, SCPR, and FPAN and RFCR. As shown in Figure 6, we have the following findings:

- (1) In terms of the Movielens dataset, our RFCR model achieves the best performance at $Q = 5$. With the increase of Q, the performance of our RFCR model has a decreasing tendency and then gets increasing. Other models either keep increasing slowly when Q increases, but all of them still cannot outperform our RFCR model on four metrics.
- (2) In terms of the BookCrossing dataset, our RFCR model achieves the best performance at $Q = 5$. With the increase of Q, the performance of our RFCR model keeps decreasing slowly. Other models keep increasing when Q increases, but all of them still cannot outperform our RFCR model in terms of the four metrics.
- (3) In terms of the Anime dataset, with the increase of Q, the performance of our RFCR model keeps increasing and achieves the best performance at $Q = 15$. It is reasonable because the Anime dataset has a larger attribute space than that in Movielens and BookCrossing dataset, and thus our RFCR model needs more turns to interact with users to find users' potential purchasing intentions and intrinsic purchasing interests. The performance of baseline models increases slowly when Q increases and all of these methods cannot outperform our RFCR model in terms of the four metrics.

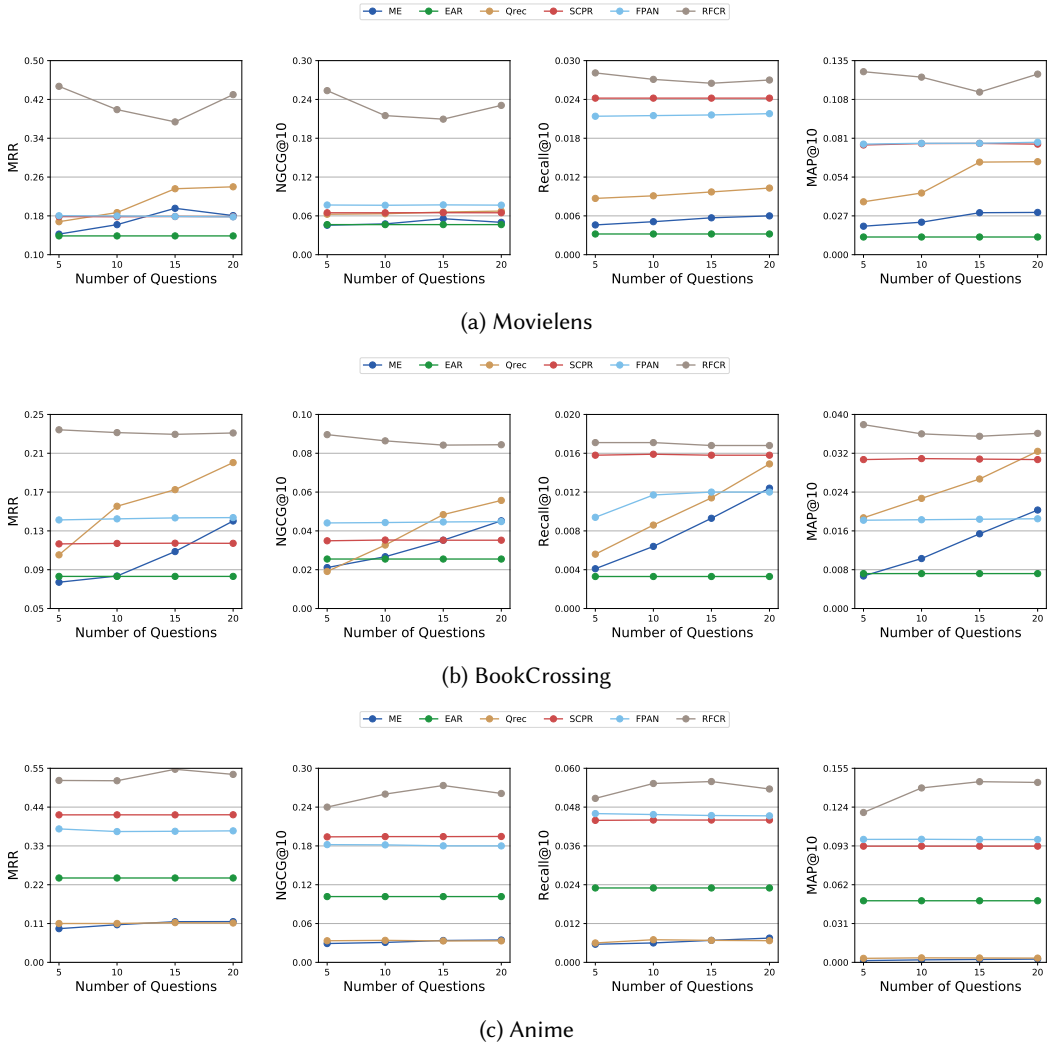


Fig. 6. Performance comparison of all methods across the different number of questions, from 5 to 20, for the three datasets, (a) Movielens, (b) BookCrossing, (c) Anime, from the top to the bottom, respectively.

Table 6 demonstrates that our RFCR model has abilities to find proper questions for users within as few turns as possible. Note that in the Movielens dataset, the performance of our RFCR model still has an increasing tendency when $Q = \{15, 20\}$, but users often interact with a CRS within 5 turns in the real-world scenarios, so we think our RFCR model has achieved the best performance at $Q = 5$. In addition, the decreasing tendency in our RFCR models is reasonable that when the questions are over the users' needs, users would give wrong feedback to a CRS or refuse to give any response, which causes that a CRS could not update user-item representations towards the right directions. Therefore, a CRS could not produce proper recommendation lists for users.

Table 9. Comparison among variants of our **RFCR** model, where **RFCR-R-L** is a variant of our **RFCR** model, which randomly selects attributes to construct questions for users in the LQS subsession; **RFCR-R-G** is a variant of our **RFCR** model, which randomly selects attributes to construct questions for users in the GQS subsession; **RFCR-W-L** is a variant of our **RFCR** model, which removes the LQS subsession and only keeps the GQS subsession in the whole conversation session (i.e., $Q_2 = Q$); **RFCR-W-G** is a variant of our **RFCR** model, which removes the GQS subsession and only keeps the LQS subsession in the whole conversation session (i.e., $Q_1 = Q$). The best performance per metric per column is marked in boldface.

	Movielens				**BookCrossing**				**Anime**			
Method	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10
RFCR	**.0281**	**.1273**	**.4470**	**.2536**	**.0171**	**.0379**	**.2342**	**.0896**	**.0507**	**.1197**	**.5155**	**.2400**
RFCR-R-L	.0189	.0849	.3067	.1708	.0150	.0331	.1958	.0686	.0461	.0965	.4138	.1968
RFCR-R-G	.0168	.0726	.2736	.1565	.0165	.0364	.2247	.0862	.0471	.1074	.4839	.2183
RFCR-W-L	.0024	.0085	.1015	.0310	.0059	.0101	.0691	.0203	.0059	.0091	.1014	.0302
RFCR-W-G	.0028	.0104	.1094	.0330	.0041	.0077	.0655	.0196	.0060	.0091	.1148	.0346

6.4 Evaluating Key Design in RFCR (RQ 4)

The key design in our proposed model is that we proposed a novel Question Search scheme to explore proper questions for users from two strategies in a CRS and fuse user-item representations obtained from two subsessions. In the LQS subsession, we utilize the directed graph to eliminate irrelevant attributes and construct local questions to capture users' intrinsic purchasing interests from historic log data. In the GQS subsession, we propose to find users' potential purchasing intentions and construct questions for users based on their preferences from the whole attribute pool. Finally, we fuse the user-item representations obtained from the LQS subsession and GQS subsession to capture more precise user preferences. In this subsection, as indicated by the analysis in § 6.1, we also set the total number of questions to 5 in the conversation session for fair comparisons.

To further verify the effectiveness of our method, we construct additional experiments by designing four variants of our RFCR model, namely RFCR-R-G, RFCR-R-L, RFCR-W-G and RFCR-W-L. Details of these variants are referred to § 5.6.

Note that we keep other components in these variant models unchanged, i.e., keeping the dimensions of the user-item representations, the splitting scheme of Q, and other hyper-parameters mentioned in § 5.4 unchanged. We report the performance of RFCR, RFCR-R-G, RFCR-R-L, RFCR-W-G and RFCR-W-L on Movielens, BookCrossing, and Anime over our four metrics.

From Table 9, we have the following discoveries:

- (1) The RFCR-R-L model performs worse than the RFCR model, especially in the Anime dataset. The Anime dataset has the most attributes than other datasets. The directed graph in LQS subsession eliminates irrelevant attributes and we construct questions for users based on the attributes from historic log data, which ensures that we can capture users' intrinsic interests in items. It validates that our scheme designed in the LQS subsession is effective.
- (2) The RFCR model outperforms the RFCR-R-G model, especially in the Movielens dataset. In the GQS subsession, we construct questions for users based on their initial beliefs from the whole attribute pool and update their beliefs according to feedback in the conversation session, which explores the users' potential interests. It validates that our scheme designed in the GQS subsession is effective.
- (3) The RFCR-W-G model and the RFCR-W-L model construct questions and produce the final user-item representations only from a single strategy, and they perform extremely poorly than the RFCR model. We can also find that both of them even work worse than RFCR-R-L and RFCR-R-G. Therefore, constructing questions from both exploitative and exploratory strategies and representation fusion are able to help improve the performance in our model.

From the above, we demonstrate that the **LQS** subsession, the **GQS** subsession, and representation fusion are effective in our RFCR.

Table 10. Statistics of the cold-start items tuples, where **proportion** indicates the proportion of cold-start items tuples in the corresponding subset (in brackets).

Dataset	#tuples (test)	proportion (test)	#tuples (test+val)	proportion (test+val)
Movielens	462	4.48%	1,395	4.61%
BookCrossing	3	0.03%	11	0.05%
Anime	169	1.47%	536	1.56%

Table 11. The results on cold-start item tuples. The best performance per metric per dataset is marked in boldface.

	Movielens				**Anime**			
Q	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10	*Recall*@10	*MAP*@10	*MRR*	*NDCG*@10
5	**.0122**	**.0537**	**.3870**	**.2159**	.0240	.0537	.4573	.2012
10	.0097	.0396	.3007	.1477	.0242	.0536	.4306	.1965
15	.0120	.0462	.3853	.2016	**.0253**	**.0592**	**.4857**	**.2221**
20	.0100	.0391	.3250	.1455	.0242	.0542	.4417	.2034

6.5 Cold Start Performance Analysis (RQ 5)

To explore whether our proposed method is effective for the cold-start user and the cold-start item problem, we extract cold-start user tuples (i.e., user-item interactions in which the user never appears in the training set) and cold-start item tuples (i.e., user-item interactions in which the item never appears in the training set) from our testing dataset. Because in the LQS subsession, we need to find attributes that are connected with the user. However, this is impossible for cold-start users who have no log data. In addition, since there are very few cold-start item tuples in the testing dataset, we also add the cold-start item tuples in validation sets for evaluation. Table 10 shows the statistics of the three datasets, where there are about 1,395 (4.61%) cold-start item tuples in the Movielens dataset, about 11 (0.05%) cold-start item tuples in the BookCrossing dataset, and about 536 (1.56%) cold-start item tuples in the Anime dataset. Since there are few cold-start item tuples in BookCrossing, we only use cold-start item tuples in Movielens and Anime to validate the ability of our RFCR model to address the cold-start problem.

The results on these two datasets are shown in Table 11, where our RFCR model still achieves high performance for cold start items and the trend of performance is consistent on the two datasets as shown in Figure 6. So, we can conclude that our RFCR model is able to tackle the cold-start recommendation problem.

6.6 Case Study on Question Search Methods (RQ6)

Our RFCR model has achieved superior performance on four metrics, and it is also explainable in constructing questions for users in CRS. Our RFCR model constructs questions for users in the conversation session from the exploitative strategy and exploratory strategy. From the exploitative strategy, we eliminate irrelevant questions for users with the directed graph and capture users' intrinsic purchasing intentions in the LQS subsession. From the exploratory strategy, we construct questions for users based on users' beliefs from the whole attribute set and could explore the user's potential new interests. These two strategies bring crystally clear question search logic, which is naturally explainable.

As shown in Figure 7, we display the whole process of constructing questions for users from the exploitative strategy and exploratory strategy. The conversation session is initialized when the user u_i enters our RFCR model. In the LQS subsession, we provide a directed graph, where nodes represent users, items, and attributes, and edges represent the connections from users to items and the connections from items to attributes. With the help of this directed graph, we can easily

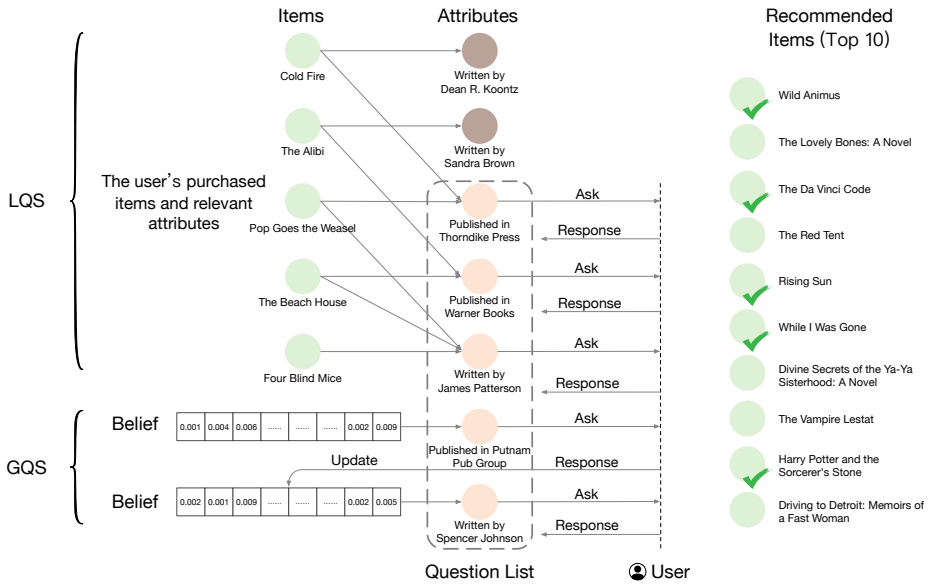


Fig. 7. A case study of our RFCR model for the book recommendation, where we construct 3 questions from the LQS subsession and 2 questions from the GQS subsession for the user. In the list of recommended items, items with green tick are ground truth items.

eliminate irrelevant attributes and find user u_i's most relevant attributes by sorting the number of paths from u_i to certain attributes. Thus, the user u_i's most interesting books should have three shared attributes: *Published in Thorndike Press*, *Published by Warner Books* and *Written by James Patterson*. In contrast, in the GQS subsession, we first construct a question based on user u_i's belief from the whole attribute pool to explore user u_i's potential purchasing intentions. Then when user u_i gives his/her response, user u_i's belief would be updated according to user u_i's feedback. Then, we utilize the updated belief to select the attribute for the next turn. We can see that when our RFCR model travels in the whole attribute pool and finally reaches user u_i potential preferred attribute in current turn (i.e., from *pulished in Putnam Pub Group* to *written by Spencer Johnson*).

7 CONCLUSION AND DISCUSSION

In this paper, we study the problem of the multi-turn conversational recommendation problem and focus on how to construct proper questions for improving the performance of recommendations and how to release the heavy burden of training policy agents with a large corpus. To address the task, we propose a novel conversational recommendation system, Representation Fusion-based Conversational Recommendation model, RFCR, where we adopt different Question Search methods to construct proper questions for users from two strategies and fuse the embeddings from two subsessions in CRS. Specifically, in the Local Question Search subsession, we use the user-item interactions and item-attribute associations to construct a graph and sort the attributes by the number of paths from users to certain attributes. Subsequently, we use the ranks as indicators to construct questions, such that the attributes can reflect the user's long-term preference. In the Global Question Search subsession, we propose a novel method to initialize the user preference, construct questions for users based on their preference and update preference according to user feedback. We then update the embeddings in two subsessions in respective turns. Finally, we fuse

these embeddings from these two subsessions and produce a recommendation. By fusing these embeddings that reflect different views of user preference, we can get a more precise representation of user preference. We conduct our experiments on the Movielens, BookCrossing, Anime datasets and adopt MaxEntropy, EAR, SCPR, Qrec, and FPAN as our baselines for comparisons. Experimental results show that RFCR presents superior performance compared to the state-of-the-art baselines.

There are still works to explore CRS in the future. First, the richer types of questions could be constructed by using other sources such as structural item properties and domain-specific informative terms. Second, more sophisticated graph mining methods, such as graph-based neural network methods, can be considered to enhance the ability to capture users' intrinsic interests. Third, we can consider more complex problem settings in a CRS, where a CRS can ask users questions such as "Which genres of movies do you like most" or "What actors do you like most?" and users in the conversation session can respond "I like it" or "I don't know" instead of "Yes" or "No". Finally, the optimal number of Q in different datasets needs to be further explored, as they are set as hyper-parameters and predefined by the exploring experiments in our paper instead of automatically optimized and found. We will attempt to improve our work and propose an adaptive method to find the optimal Q in different datasets for reducing the burden of tuning hyper-parameters and avoiding falling into sub-optimal situations.

ACKNOWLEDGMENTS

We thank the reviewers for their helpful and constructive suggestions. This work is supported by the National Natural Science Foundation of China (grant No. 61906219) and the MBZUAI-WIS Joint Program for Artificial Intelligence Research. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. *SIGIR*, 475–484.
[2] Rodrigo Alves, Antoine Ledent, and Marius Kloft. 2021. Burst-induced Multi-Armed Bandit for Learning Recommendation. In *Recsys*. 292–301.
[3] Martin Anthony, Peter L Bartlett, Peter L Bartlett, et al. 1999. *Neural network learning: Theoretical foundations*.
[4] Robert Bell, Yehuda Koren, and Chris Volinsky. 2007. Modeling relationships at multiple scales to improve accuracy of large recommender systems. *KDD*, 95–104.
[5] Keping Bi, Qingyao Ai, Yongfeng Zhang, and Bruce W. Croft. 2019. Conversational Product Search Based on Negative Feedback. *CIKM*, 359–368.
[6] Guanzheng Chen, Jinyuan Fang, Zaiqiao Meng, Qiang Zhang, and Shangsong Liang. 2022. Multi-Relational Graph Representation Learning with Bayesian Gaussian Process Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 5530–5538.
[7] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. *EMNLP/IJCNLP*, 1803–1813.
[8] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and H. Ed Chi. 2018. Q&R: A Two-Stage Approach toward Interactive Recommendation. *SIGKDD*, 139–148.
[9] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. *SIGKDD*, 815–824.
[10] Chongming Gao, Wenqiang Lei, Xiangnan He, de Maarten Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *arXiv preprint arXiv:2101.09459* (2021).
[11] Yingqiang Ge, Shuya Zhao, Honglu Zhou, Changhua Pei, Fei Sun, Wenwu Ou, and Yongfeng Zhang. 2020. Understanding echo chambers in e-commerce recommender systems. *SIGIR*, 2261–2270.
[12] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. 2017. On context-dependent clustering of bandits. *ICML*, 1253–1262.
[13] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical User Profiling for E-commerce Recommender Systems. *WSDM*, 223–231.

[14] Maxwell F. Harper and A. Joseph Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* (2016), 19:1–19:19.

[15] Claudia Hauff, Julia Kiseleva, Mark Sanderson, Hamed Zamani, and Yongfeng Zhang. 2021. Conversational Search and Recommendation: Introduction to the Special Issue. , 6 pages.

[16] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. *SIGIR*, 355–364.

[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[18] Guo Huifeng, Tang Ruiming, Ye Yunming, Li Zhenguo, and He Xiuqiang. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. *IJCAI*, 1725–1731.

[19] Andrea Iovine, Pasquale Lops, Fedelucio Narducci, Marco de Gemmis, and Giovanni Semeraro. 2021. Improving preference elicitation in a conversational recommender system with active learning strategies. In *Annual ACM Symposium on Applied Computing*. 1375–1382.

[20] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2020. A survey on conversational recommender systems. *arXiv preprint arXiv:2004.00646* (2020).

[21] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A Survey on Conversational Recommender Systems. *CSUR* (2021), 1–36.

[22] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* (2002), 422–446.

[23] Edwin T Jaynes. 1982. On the rationale of maximum-entropy methods. *Proc. IEEE* 70, 9 (1982), 939–952.

[24] Shaojie Jiang, Pengjie Ren, Christof Monz, and Maarten de Rijke. 2019. Improving neural response diversity with frequency-aware cross-entropy loss. *WWW*, 2879–2885.

[25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. *ICDM*, 197–206.

[26] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR (Poster)*.

[27] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* (2009), 30–37.

[28] Wenqiang Lei, Chongming Gao, and Maarten de Rijke. 2021. RecSys 2021 Tutorial on Conversational Recommendation: Formulation, Methods, and Evaluation. *Recsys*, 842–844.

[29] Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2020. Conversational recommendation: Formulation, methods, and evaluation. *SIGIR*, 2425–2428.

[30] 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. *WSDM*, 304–312.

[31] 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. *SIGKDD*, 2073–2083.

[32] Dongdong Li, Zhaochun Ren, Pengjie Ren, Zhumin Chen, Miao Fan, Jun Ma, and Maarten de Rijke. 2021. Semi-Supervised Variational Reasoning for Medical Dialogue Generation. *SIGIR* (2021).

[33] Guohui Li, Qi Chen, Bolong Zheng, Nguyen Quoc Viet Hung, Pan Zhou, and Guanfeng Liu. 2020. Time-aspect-sentiment Recommendation Models Based on Novel Similarity Measure Methods. *TWEB* (2020), 1–26.

[34] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

[35] Raymond Li, Ebrahimi Samira Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. *NeurIPS*, 9748–9758.

[36] 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. *TOIS* (2021).

[37] Shangsong Liang, Yupeng Luo, and Zaiqiao Meng. 2021. Profiling users for question answering communities via flow-based constrained co-embedding model. *ACM Transactions on Information Systems (TOIS)* 40, 2 (2021), 1–38.

[38] Shangsong Liang, Shaowei Tang, Zaiqiao Meng, and Qiang Zhang. 2022. Cross-temporal snapshot alignment for dynamic networks. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–15.

[39] Bang Liu, Hanlin Zhang, Linglong Kong, and Di Niu. 2021. Factorizing Historical User Actions for Next-Day Purchase Prediction. *TWEB* (2021), 1–26.

[40] Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Towards Conversational Recommendation over Multi-Type Dialogs. *ACL*, 1036–1049.

[41] Robert Nowak. 2008. Generalized binary search. *Annual Allerton Conference on Communication, Control, and Computing*, 568–574.

[42] Yaoxin Pan, Shangsong Liang, Jiaxin Ren, Zaiqiao Meng, and Qiang Zhang. 2021. Personalized, sequential, attentive, metric-aware product search. *ACM Transactions on Information Systems (TOIS)* 40, 2 (2021), 1–29.

[43] Zhiqiang Pan, Fei Cai, Wanyu Chen, and Honghui Chen. 2021. Graph Co-Attentive Session-based Recommendation. *TOIS* (2021).
[44] Jiahuan Pei, Pengjie Ren, and Maarten de Rijke. 2021. A cooperative memory network for personalized task-oriented dialogue systems with incomplete user profiles. *WWW*, 1552–1561.
[45] Dhanya Pramod and Prafulla Bafna. 2022. Conversational Recommender Systems Techniques, Tools, Acceptance, and Adoption: A State of the Art Review. *Expert Systems with Applications* (2022), 1–18.
[46] Bilih Priyogi. 2019. Preference elicitation strategy for conversational recommender system. *WSDM*, 824–825.
[47] Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. 2019. Coached Conversational Preference Elicitation: A Case Study in Understanding Movie Preferences. *ACL*, 353–360.
[48] Matěj Račinský. 2018. MyAnimeList Dataset. *Kaggle*.
[49] Xuhui Ren, Hongzhi Yin, Tong Chen, Hao Wang, Zi Huang, and Kai Zheng. 2021. Learning to Ask Appropriate Questions in Conversational Recommendation. *SIGIR* (2021).
[50] Xuhui Ren, Hongzhi Yin, Tong Chen, Hao Wang, Nguyen Quoc Viet Hung, Zi Huang, and Xiangliang Zhang. 2020. Crsal: Conversational recommender systems with adversarial learning. *TOIS* (2020), 1–40.
[51] Steffen Rendle. 2010. Factorization Machines. *ICDM*, 995–1000.
[52] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. 291–324.
[53] Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. 2018. Preference elicitation as an optimization problem. *Recsys*, 172–180.
[54] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *JMLR* (2005).
[55] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. *SIGIR*, 235–244.
[56] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2017. Factorization bandits for interactive recommendation. *AAAI*.
[57] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. *SIGIR*, 165–174.
[58] Bin Wu, Zaiqiao Meng, Qiang Zhang, and Shangsong Liang. 2022. Meta-Learning Helps Personalized Product Search. In *Proceedings of the ACM Web Conference 2022*. 2277–2287.
[59] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. *SIGIR*, 726–735.
[60] Qingyun Wu, Naveen Iyer, and Hongning Wang. 2018. Learning Contextual Bandits in a Non-stationary Environment. *SIGIR*, 495–504.
[61] 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. *WSDM*, 364–372.
[62] Tong Yu, Yilin Shen, and Hongxia Jin. 2019. A visual dialog augmented interactive recommender system. *KDD*, 157–165.
[63] Xiaofeng Yuan, Lixin Han, Subin Qian, Licai Zhu, Jun Zhu, and Hong Yan. 2021. Preliminary data-based matrix factorization approach for recommendation. *Information Processing & Management* (2021).
[64] Dave Zachariah, Martin Sundin, Magnus Jansson, and Saikat Chatterjee. 2012. Alternating least-squares for low-rank matrix reconstruction. *IEEE Signal Processing Letters*, 231–234.
[65] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating clarifying questions for information retrieval. *WWW*, 418–428.
[66] Xiaoying Zhang, Hong Xie, Hang Li, and C.S. John Lui. 2020. Conversational Contextual Bandit: Algorithm and Application. *WWW*, 662–672.
[67] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and Bruce W. Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. *CIKM*, 177–186.
[68] Chunyi Zhou, Yuanyuan Jin, Xiaoling Wang, and Yingjie Zhang. 2020. Conversational Music Recommendation based on Bandits. *ICKG*, 41–48.
[69] Kun Zhou, Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving Conversational Recommender Systems via Knowledge Graph based Semantic Fusion. *SIGKDD*, 1006–1014.
[70] Tianyu Zhu, Leilei Sun, and Guoqing Chen. 2021. Graph-based Embedding Smoothing for Sequential Recommendation. *TKDE* (2021).
[71] Yadong Zhu, Xiliang Wang, Qing Li, Tianjun Yao, and Shangsong Liang. 2021. Botspot++: A hierarchical deep ensemble model for bots install fraud detection in mobile advertising. *ACM Transactions on Information Systems (TOIS)* 40, 3 (2021), 1–28.
[72] Jie Zou and Evangelos Kanoulas. 2019. Learning to Ask: Question-based Sequential Bayesian Product Search. *CIKM*, 369–378.

[73] Jie Zou, Dan Li, and Evangelos Kanoulas. 2018. Technology Assisted Reviews: Finding the Last Few Relevant Documents by Asking Yes/No Questions to Reviewers. *SIGIR*, 949–952.
[74] Jie Zou, Chen Yifan, and Kanoulas Evangelos. 2020. Towards Question-based Recommender Systems. *SIGIR*, 881–890.