Interactive Path Reasoning on Graph for Conversational Recommendation

Wenqiang Lei[1], Gangyi Zhang[2], Xiangnan He[2*], Yisong Miao[1], Xiang Wang[1], Liang Chen[3], Tat-Seng Chua[1]
[1]National University of Singapore, [2]University of Science and Technology of China, [3]Sun Yat-Sen University
wenqianglei@gmail.com,gangyi.zhang@outlook.com,xiangnanhe@gmail.com,miaoyisong@gmail.com
xiangwang@u.nus.edu,chenliang6@mail.sysu.edu.cn,chuats@comp.nus.edu.sg

ABSTRACT

Traditional recommendation systems estimate user preference on items from past interaction history, thus suffering from the limitations of obtaining fine-grained and dynamic user preference. Conversational recommendation system (CRS) brings revolutions to those limitations by enabling the system to directly ask users about their preferred attributes on items. However, existing CRS methods do not make full use of such advantage — they only use the attribute feedback in rather implicit ways such as updating the latent user representation. In this paper, we propose **C**onversational **P**ath **R**easoning (**CPR**), a generic framework that models conversational recommendation as an interactive path reasoning problem on a graph. It walks through the attribute vertices by following user feedback, utilizing the user preferred attributes in an explicit way. By leveraging on the graph structure, CPR is able to prune off many irrelevant candidate attributes, leading to better chance of hitting user preferred attributes. To demonstrate how CPR works, we propose a simple yet effective instantiation named **SCPR** (**S**imple **CPR**). We perform empirical studies on the multi-round conversational recommendation scenario, the most realistic CRS setting so far that considers multiple rounds of asking attributes and recommending items. Through extensive experiments on two datasets Yelp and LastFM, we validate the effectiveness of our SCPR, which significantly outperforms the state-of-the-art CRS methods EAR [13] and CRM [24]. In particular, we find that the more attributes there are, the more advantages our method can achieve.

CCS CONCEPTS

• **Information systems → Users and interactive retrieval**; **Recommender systems**; *Personalization*; • **Human-centered computing** → *Interactive systems and tools.*

KEYWORDS

Conversational Recommendation; Interactive Recommendation; Recommender System; Dialogue System

ACM Reference Format:
Wenqiang Lei[1], Gangyi Zhang[2], Xiangnan He[2*], Yisong Miao[1], Xiang Wang[1], Liang Chen[3], Tat-Seng Chua[1]. 2020. Interactive Path Reasoning on Graph for Conversational Recommendation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403258

1 INTRODUCTION

Personalized recommendation systems have been standard fixtures in many scenarios like E-commerce (e.g., Amazon) and content sharing platforms (e.g., YouTube). They traditionally conduct recommendations by inferring user preference on items from their historical actions [10, 20]. While proven to be a success, traditional methods suffer from the intrinsic limitation of passively acquiring user feedback in the process of making recommendations. Such information asymmetry makes it hard to obtain dynamic and fine-grained user preference, preventing the system to provide accurate and explainable recommendation service.

The recently emerging conversational recommendation system (CRS) brings revolutions to the aforementioned limitation. CRS is envisioned as the deep composition of a conversational system and a recommendation system [13]. It makes recommendations when interacting with users using natural languages and can proactively ask a user whether he/she likes an item attribute or not. As such, CRS has the natural advantage of conducting dynamic and explainable recommendation by utilizing the user's preferred attributes as interpretable reasons. However, existing works only utilize attribute feedback implicitly by mapping attributes into a latent space, which we believe does not make full use of the advantage of attribute feedback. For example, Bi et al. [1], Zhang et al. [33] update the opaque user embedding once obtaining the user feedback on an attribute. Lei et al. [13] feed the preferred attribute into a variant of factorization machine [20] to score items in the latent space. Sun and Zhang [24] feed the user attribute preference to a policy network, which is trained to decide the next action — whether to make recommendations or ask an attribute.

The key hypothesis of this work is that, a more explicit way of utilizing the attribute preference can better carry forward the advantages of CRS — being more accurate and explainable. To this end, we propose a novel conversational recommendation framework called **C**onversational **P**ath **R**easoning (CPR). Inspired by the recent success of graph-based recommendation [25], we model conversational recommendation as the process of finding a path in user-item-attribute graph interactively. Figure 1 shows an illustrative example. The vertices in the right graph represent users, items

[*] Xiangnan He is the Corresponding Author.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403258

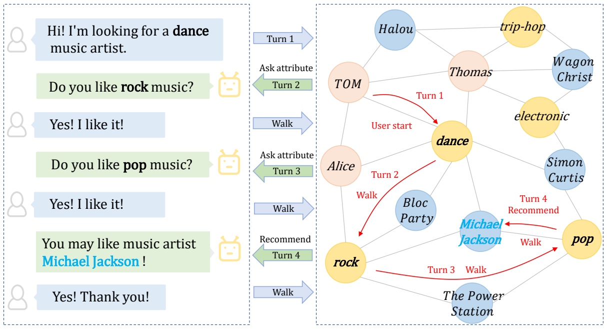


Figure 1: An illustration of interactive path reasoning in CPR. As the convention of this paper, light orange, light blue, and light gold vertices represents the user, attribute and items respectively. For example, the artiest *Michael Jackson* is an item and and the attributes are *rock, dance* etc.

and attributes as well as other relevant entities. An edge between two vertices represent their relation, for example, a user-item edge indicates that the user has interacted with the item, and a user-attribute edge indicates that the user has affirmed an attribute in a conversation session. A conversation session in our CPR is expressed as a walking in the graph. It starts from the user vertex, and travels in the graph with the goal to reach one or multiple item vertices the user likes as the destination. Note that the walking is navigated by users through conversation. This means, at each step, a system needs to interact with the user to find out which vertex to go and takes actions according to user's response.

We now go through an example in Figure 1 to better understand the process. A user *TOM* is seeking a recommendation of music artists. The walking starts from the user vertex ("*TOM*"), and the session is initialized by the user-specified attribute ("*dance*"). Accordingly, the system makes its first step from "*TOM*" to "*dance*". Afterwards, the system identifies an *adjacent attribute* (*c.f.* Sec 4.1) vertex on the graph to consult the user, or recommendation a list of items. If the user confirms his preference to the asked attribute, the system will transit to that attribute vertex. However, if the user rejects the attribute, or rejects a recommendation, the system will stay at the same vertex and consult the user for another attribute. The session will repeat such cycle multiple times until the recommended items are accepted by the user[1].

The proposed CPR framework, as a new angle of conducting conversational recommendation, conceptually brings several merits to the development of CRS:

1. It is crystally explainable. It models conversational recommendation as an interactive path reasoning problem on the graph, with each step confirmed by the user. Thus, the resultant path is the correct reason for the recommendation. This makes better use of the fine-grained attribute preference than existing methods that only model attribute preference in latent space such as [13].
2. It facilitates the exploitation of the abundant information by introducing the graph structure. By limiting the candidate attributes to ask as adjacent attributes of the current vertex, the candidate space is largely reduced, leading to a significant advantage compared with existing CRS methods like [13, 24] that treat almost all attributes as the candidates.
3. It is an aesthetically appealing framework which demonstrates the natural combination and mutual promotion of conversation system and recommendation system. On one hand, the path walking over the graph provides a natural dialogue state tracking for conversation system, and it is believed to be efficient to make the conversation more logically coherent [12, 14]; on the other hand, being able to directly solicit attribute feedback from the user, the conversation provides a shortcut to prune off searching branches in the graph.

To validate the effectiveness of CPR, we provide a simple yet effective implementation called **SCPR** (**S**imple **CPR**), targeting at the multi-round conversational recommendation (MCR) scenario (*c.f.* Sec 3). We conduct experiments on the Yelp and LastFM datasets, comparing SCPR with state-of-the-art CRS methods [13, 24] which also use the information of user, item and attribute but does not use graph. We analyze the properties of each method under different settings, including different types of questions (binary and enumerated) and different granularity of attributes. We find that SCPR outperforms existing methods on recommendation success rate, especially in the settings where the attribute space is larger.

In summary, our contributions are two-folds:

- We propose the CPR framework to model conversational recommendation as a path reasoning problem on a heterogeneous graph which provides a new angle of building CRS. To the best of our knowledge, it is the first time to introduce graph-based reasoning to multi-round conversational recommendation.
- To demonstrate the effectiveness of CPR, we provide a simple instantiation SCPR, which outperforms existing methods in various settings. We find that, the larger attribute space is, the more improvements our model can achieve.

2 RELATED WORK

The success of a recommendation system hinges on offering the relevant items of user interest accurately and timely. At beginning, recommendation systems are largely built on the collaborative filtering hypothesis to infer a distributed representation of the user profile. Representative models include matrix factorization [11] and factorization machines [9, 20]. However, by nature, these approaches suffer from two intrinsic problems. The first one is the inability of capturing user dynamic preferences with the strict assumption that a user's interest is static over the long-term horizon [23]. The second problem is the weak explainability as the user preference representation is only a continuous vector. Later works try to introduce Markov models [21] and multi-arm bandit methods [28] to solve the dynamic problem but the explainability still remains to be unsatisfactory.

Recently, **Graph-based recommendation methods** attract increasing research attention. One line of research leverages on the better expressiveness of the graph. They either explore implicit properties like collaborative signals [25, 35] from the global connectivities, or focus on yielding better representations of user/items by incorporating latent network embeddings [30]. Another line of

[1] In our descriptions on graphs, we sometime directly use the word *item*, *attribute* or *user* to refer to their corresponding vertices for simplicity.

Table 1: Main notations used in the paper.

u, v, p	User, item, and attribute
P	An *active* attribute path in the graph
aa_t	An adjacent attribute of the attribute p_t
$\mathcal{AA}_t$	The set of adjacent attributes of the attribute p_t
$\mathcal{P}_u$	The set of attributes confirmed by u in a session
$\mathcal{P}_{cand}$	The set of candidate attributes
$\mathcal{V}_p$	The set of items that contain the attribute p
$\mathcal{V}_{cand}$	The set of candidate items;
a	The action of CPR, either a_{ask} or a_{rec}

work leverages on the explainability of the graph, modeling recommendation as a path reasoning problem on the graph. They aim to find a path from a user vertex to the target item, and use the resultant path as the recommendation reason [26, 29]. While being explainable, such methods suffer from two problems: 1) they are still static models which intrinsically cannot capture the preference dynamics, and 2) the modeling complexity is high such that pruning becomes a critical step [29].

Conversational recommendation system (CRS) becomes an appealing solution to both the dynamic preference and weak explainability problems as it dynamically gets user explicit feedback. As an emerging topic, various problems under different settings have been explored [5–7, 15–17, 19, 22, 24, 31–34], such as natural language understanding and generation [5, 15], multi-model and multi-media [17], monitoring user feedback on viewing, clicking and commenting [31], and attribute prediction [19].

We believe that how to dynamically ask attribute questions and make recommendations upon attribute answer is the key at current stage of conversational recommendation. As such, we consider the system asking user preference on attributes and making recommendation based on those attributes in a multi-turn basis. As discussed in Section 1, main works [1, 13, 24, 33] along this line do not use attributes explicitly. We argue more explicitly utilizing the attribute would better carry forward the advantage of conversational recommendation. Therefore, this paper makes a key contribution to introduce graph to increase the explainability.

3 MULTI-ROUND CONVERSATIONAL RECOMMENDATION SCENARIO

As conversational recommendation is an emerging research topic, various settings have been explored in recently. This paper follows the *multi-round conversational recommendation* (MCR) scenario since it is the most realistic setting in research so far [13]. In a MCR setting, a CRS is free to ask attributes or make recommendation multiple times. We use a *round* to emphasize one trial of recommendation. This is in contrast to the *single-round conversational recommendation* as adopted by [24] where the system asks attribute multiple times followed by making recommendation only once, after which the conversation session ends regardless of whether the recommendation succeeds. The multi-round setting is more challenging than the *single-round* one as a CRS has more freedom to take actions which makes the policy space more complex.

Specifically, an item v is associated with a set of attributes $\mathcal{P}_v$. The attributes broadly cover various descriptions as long as it can describe certain properties of an item. For example, in the music artist recommendation domain (e.g., in the lastFM dataset), an item is a music artist and the attribute may be descriptions like *Jazz*, *Classic*, *Energetic*, *Peaceful* etc. The items and attributes are provided by the dataset. During a conversation session, a CRS obtains the user's fine-grained preference by asking whether he likes particular attributes. Based on such conversations, a CRS aims to provide accurate recommendations in the shortest conversational turns.

Algorithm 1 The MCR Scenario

```
Input: user $u$, all attributes $\mathcal{P}$, all items $\mathcal{V}$, the number of items
    to recommend $k$, the maximum number of turns $T$;
Output: recommendation result: success or fail;
1:  User $u$ specifies an attribute $p_0$;
2:  Update: $\mathcal{P}_u = \{p_0\}$; $\mathcal{P}_{cand} = \mathcal{P} \setminus p_0$; $\mathcal{V}_{cand} = \mathcal{V}_{p_0}$
3:  for turn $t = 1, 2, 3...T$ do
4:      Select an action $a$
5:      if $a == a_{ask}$ then
6:          Select the top attribute $p$ from $\mathcal{P}_{cand}$
7:          if $u$ accepts $p_t$ then
8:              Update: $\mathcal{P}_u = \mathcal{P}_u \cup p$; $\mathcal{V}_{cand} = \mathcal{V}_{cand} \cap \mathcal{V}_p$
9:          Update: $\mathcal{P}_{cand} = \mathcal{P}_{cand} \setminus p$
10:     else[$a == a_{rec}$]
11:         Select the top-$k$ items $\mathcal{V}_k$ from $\mathcal{V}_{cand}$
12:         if User accepts $\mathcal{V}_k$ then
13:             Recommendation succeeds; Exit.
14:         else[User rejects $\mathcal{V}_k$]
15:             Update: $\mathcal{V}_{cand} = \mathcal{V}_{cand} \setminus \mathcal{V}_k$
16: Recommendation fails; Exit.
```

A conversation session starts on the user side, which initializes the attribute p_0 by specifying an attribute the user likes (e.g., I like some *dance* music). Next, the CRS is free to ask his preference on an attribute selected from the candidate attribute set $\mathcal{P}_{cand}$ or recommend items from the candidate item set $\mathcal{V}_{cand}$. Then, the user needs to give feedback accordingly, either accepting or rejecting them. The CRS makes use of such feedback from the user — if the user accepts the asked attribute, the CRS puts it in the preferred attribute set $\mathcal{P}_u$ and removes it from $\mathcal{P}_{cand}$. Then the CRS updates $\mathcal{V}_{cand}$ to $\mathcal{V}_{cand} \cap \mathcal{V}_p$, representing the items containing all attribute confirmed by the user in the session. $\mathcal{V}_p$ denote the items containing the attribute p; if he rejects the asked attribute, the CRS removes it from $\mathcal{P}_{cand}$; if he rejects the recommended items, the CRS removes them from $\mathcal{V}_{cand}$. Based on the updated sets, the CRS takes the next action, i.e., *asking* or *recommending*, and repeats the above process. The conversation session ends until the CRS hits the user preferred items or reaches the maximum number of turns T. This process is detailed in Algorithm 1.

Following Lei et al. [13], it is noticeable that the above MCR scenario makes two assumptions. (1) It assumes that the user clearly expresses his preferences by specifying attributes without any reservations, and the items containing the preferred attributes are enough in the dataset. Given this assumption, the CRS takes the attributes accepted by the user as a strong indicator. For example, it only considers all items containing all attributes he accepts (line 2 and line 8 in Algorithm 1). This is because the items that contain all the preferred attributes have higher priority than the items do not. Since such higher-prioritized items are enough, ignoring the other candidate items is a reasonable simplification to this problem. (2)

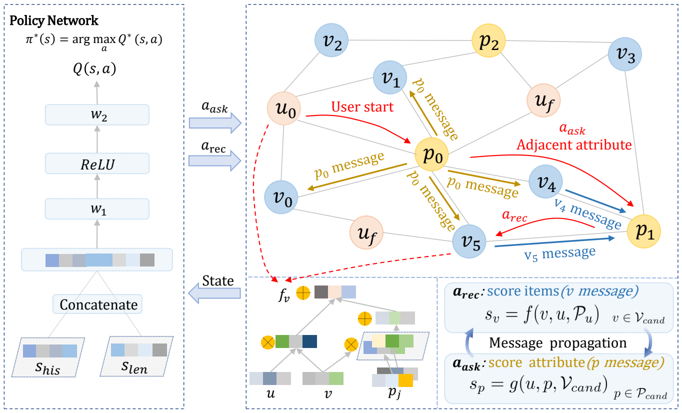


Figure 2: CPR framework overview. It starts from the user u_0 and walks over adjacent attributes, forming a path (the red arrows) and eventually leading to the desired item. The policy network (left side) determines whether to ask an attribute or recommend items in a turn. Two reasoning functions f and g score attributes and items, respectively.

It assumes that the CRS does not handle strong negative feedback. This means, if a user rejects the asked attribute, the CRS does not distinguish whether the user *does not care* it or *hates* it. It is because such negative feedback is hard to obtain in current data, making it difficult to simulate in experimental surroundings. Therefore, the CRS equally treats all rejected attributes as *does not care* and only removes the attributes from the candidate set without further actions like removing all items that contain the rejected attributes.

In this scenario, Lei et al. [13] distills several key research problems, such as: (1) Which items to recommend? (2) Which attribute to ask? (3) When to ask attributes and when to make recommendations? We next articulate how our method conceptually brings benefits to address these questions.

4 PROPOSED METHODS

We first propose *Conversational Path Reasoning* (CPR), a general solution framework for graph-based conversational recommendation. We then introduce a simple yet effective instantiation SCPR to demonstrate how it works.

4.1 CPR Framework

A graph uses vertices to represent entities and edges to represent the relationships between entities. Specifically, a graph G is defined as a set of triplets $\{(h, r, t)\}$, indicating a certain relation r exists between the head entity h and the tail entity t. In this paper, we consider the graph containing three types of entities, namely, user u, item v, and attribute p. The relations between each types of entities can vary a lot depending on specific datasets (*c.f.* Table 4 in Appendix A). For example, in Figure 2, the edge between the u_0 and p_0 means the u_0 has specified his preference on attribute p_0 in his static profile; the edge between u_0 and v_0 indicate the user u_0 has interacted with v_0. Note that, in this paper, we do not specifically model different semantics of relations, and only care care whether there is an edge between two vertices for simplicity. In addition, the item, attributes and user information and their relations are also used by existing conversational recommendations systems [13, 24]. The difference is that, our CPR organizes such three types of information in graph and leverages on the advantages of graph structure to conduct conversational recommendation.

In the MCR scenario, the system treats attributes as the preference feedback. To explicitly utilize these feedback, CPR performs the walking (i.e., reasoning) over the attribute vertices. Specifically, CPR maintains an active path P, comprising the attributes confirmed by a user (i.e., all attributes in $\mathcal{P}_u$) in the chronological order, and exploring on the graph for the next attribute vertex to walk. Note that, (1) CPR does not visit the attributes that have been visited before and does not consider the directions of edges.(2) The walking in CPR differs from existing work of graph-based recommendation [26, 29], which performs the walking over all types of vertices. We believe that restricting walking on attributes as in CPR brings two benefits. First, it emphasizes the importance of the attributes as explicit reasons for recommendation. Second, it makes the walking process more concise, eliminating the uncertainty in an unnecessarily long reasoning path which might lead to error accumulation [29].

Now, we move to the detailed walking process in CPR. Assume the current active path is $P = p_0, p_1, p_2, ..., p_t$. The system stays at p_t and is going to find the next attribute vertex to walk. This process can be decomposed into three steps: *reasoning*, *consultation* and *transition*.

4.1.1 *Reasoning.*

This is the beginning of a turn. It is triggered when an attribute is initialized or confirmed by the user. In this step, CPR scores items and attributes, solving the problem of *which items to recommend* and *which attribute to ask*. In the context of MCR, CPR makes the key contribution of formalizing the scoring as message propagation on the graph. Because the scoring of attributes and items are interdependent, we adopt an alternating optimization strategy to optimize them in an asynchronous manner which has been proven to be effective [36].

First, the alternating optimization propagates messages from attributes to items to score the items (the light gold arrows in Figure 2). Specifically, all attributes in the path P (i.e., $\forall p_i \in \mathcal{P}_u$) together with the user vertex u propagate messages to candidate items in $\mathcal{V}_{cand}$ (from Section 3, we know that $\mathcal{V}_{cand}$ actually corresponds to the vertices directly connecting all $\mathcal{P}_u$). As an example, in Figure 2, when a user initializes his preferred attribute p_0 (i.e., $P = p_0$), the CPR propagates messages from p_0 to its directly connected items (i.e., v_0, v_1, v_4, v_5) to score these items. The scoring function for each item can be any implementation of traditional recommender models, abstracted as

$$s_v = f(v, u, \mathcal{P}_u), \tag{1}$$

where s_v is a scalar indicating the recommendation score of item v in the current conversation session, and $\mathcal{P}_u$ denotes the attributes confirmed by u in the session.

Second, the candidate items in turn propagate messages to the candidate attributes (the light blue arrows in Figure 2). The idea is that, with updated scores (i.e., s_v) calculated in the first step, the items provide additional information to find proper attributes to consult the user. For example, the attributes that can reduce the uncertainty in item scoring. Specifically, CPR leverages on the natural constraint of graph structure, considering only the transition to the **adjacent attributes** — if the shortest path between attribute p_t and aa_t does not contain any other attribute, then aa_t is the *adjacent attribute* of p_t. For example, in the graph of Figure 2, both p_1 and p_2 are the adjacent attribute of p_0. Formally, in CPR, the candidate attribute set $\mathcal{P}_{cand} = \mathcal{AA}_t \setminus (\mathcal{P}_u \cup \mathcal{P}_{rej})$, where $\mathcal{AA}_t$ stores all adjacent attributes of p_t and $\mathcal{P}_{rej}$ is the attributes rejected by the user. Finally, for a candidate attribute $p \in \mathcal{P}_{cand}$, its score is calculated by propagating messages from the candidate items $\mathcal{V}_{cand}$:

$$s_p = g(u, p, \mathcal{V}_{cand}), \tag{2}$$

This adjacent attribute constraint brings two benefits. (1) In terms of recommendation, it significantly reduces the search space for selecting which attribute to ask. Note that state-of-the-art conversational recommendation systems like EAR [13] and CRM [24] treat the whole attribute set $\mathcal{P}$ as the candidate space, increasing the difficulty to learn a good decision function. (2) In terms of conversation, constraining the adjacent attributes makes the dialogue more coherent. In linguistics, the closer the two entities are in any two adjacent utterances, the more coherent the conversation will be [8].

4.1.2 *Consultation.*

Once a reasoning step is completed, CPR moves to the *consultation* step. The purpose of this step is to decide *whether to ask an attribute* or *to recommend items*, with the goal of achieving successful recommendations in fewest turns. We address it as a reinforcement learning (RL) problem. Specifically, a policy function $\pi(\mathbf{s})$ is expected to make the decision based on the global dialogue state $\mathbf{s}$, which can include any information useful for successful recommendation, such as the dialogue history, the information of candidate items.The output action space of the policy function contains two choices: a_{ask} or a_{rec}, indicating whether to perform top-k recommendations or to ask an attribute in this turn. If the RL decision is a_{ask}, we directly take highest-scored attribute from $\mathcal{P}_{cand}$, where the score is s_p as defined in Eq. (2). Otherwise, we recommend top-k items from V_{cand} according to the score of s_v, which is defined in Eq. (1).

It is worth mentioning that our design of RL here reflects another major difference with existing conversational recommendation systems EAR [13] and CRM [24]. Although they also learn policy networks with RL, their policy is to decide *which attribute to ask*, rather than our choice of *whether to ask attribute*. Which means, the size of their action space is $|\mathcal{P}| + 1$, where $|\mathcal{P}|$ denotes the number of attributes. This greatly increases the difficulty to learn the policy well, especially for a large $|\mathcal{P}|$, since RL is notoriously difficult to train when the action space is large [4]. In contrast, the action space of our RL is of size 2, being much easier to train.

4.1.3 *Transition.*

The *transition* step will be triggered after the user confirms an asked attribute p_t. CPR first performs walking from the last confirmed attribute p_{t-1} to p_t, forming an extended path $P = p_0, p_1, ...p_{t-1}, p_t$. Then, we add p_t to the preferred attribute set $\mathcal{P}_u$. Accordingly, the candidate attribute set is updated by $\mathcal{P}_{cand} = \mathcal{AA}_t \setminus (\mathcal{P}_u \cup \mathcal{P}_{rej})$, and the candidate item set V_{cand} is updated by keeping the items that directly link to all attributes in the updated $\mathcal{P}_u$. Note that, if an attribute is rejected by the user, we just remove it from the candidate attribute set without vertex transition (line 9 of Algorithm 1). After the transition, our CPR starts the next conversation turn, repeating the same *reasoning–consultation–transition* process [2] .

4.2 SCPR Model

To materialize the CPR framework, we need to specify functions $f(v, u, \mathcal{P}_u)$, $g(u, p, \mathcal{V}_{cand})$ and $\pi(\mathbf{s})$. We here provide a simple implementation SCPR, adapting some designs from EAR [13] — a latest conversational recommendation system.

4.2.1 *Reasoning - Item Scoring.*

In the reasoning stage, $f(v, u, \mathcal{P}_u)$ scores the item v by propagating messages from the user-preferred attributes. We use the inner product between two vertex embeddings as the message, same as the FM variant used in EAR:

$$f(v, u, \mathcal{P}_u) = \mathbf{u}^T\mathbf{v} + \sum_{p \in \mathcal{P}_u} \mathbf{v}^T\mathbf{p}, \tag{3}$$

where $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{p}$ denote the embedding of the user u and item v and attribute p, respectively. The first term models the message propagated from the user to the item, and the second term models the messages propagated from user-preferred attributes to the item. These embeddings are randomly initialized and trained offline with the goal of scoring the interacted items higher than the non-interacted ones. The training objective is a multi-task pairwise loss, which follows EAR and we leave the details to Appendix B.

4.2.2 *Reasoning - Attribute Scoring.*

Another function of the reasoning step is to decide which attribute is worth asking according to the current system state. An expected strategy is to find the one that can better eliminate the uncertainty of items. As information entropy has proven to be an effective method of uncertainty estimation [27], we implement the $g(u, p, \mathcal{V}_{cand})$ function as information entropy but adapt it to a weighted way:

$$g(u, p, \mathcal{V}_{cand}) = -\text{prob}(p) \cdot \log_2(\text{prob}(p)),$$
$$\text{prob}(p) = \frac{\sum\limits_{v \in \mathcal{V}_{cand} \cap \mathcal{V}_p} \sigma(s_v)}{\sum\limits_{v \in \mathcal{V}_{cand}} \sigma(s_v)}, \tag{4}$$

where σ is the sigmoid function to normalize the item score s_v to $(0, 1)$, $\mathcal{V}_{cand}$ denotes the candidate items, and $\mathcal{V}_p$ denotes the items that include the attribute p. Different from the standard entropy which treats each item equally, our weighted entropy employed here assign higher weights to the important items (i.e., the items in $\mathcal{V}_p$ and scored higher) in attribute scoring. If there is no message propagated to an attribute, we define its entropy to be 0. Note that, in this implementation, we do not consider user u for calculating g for simplicity. It does not not mean we don't value the importance of u in deciding attribute. We leave the exploration of incorporating u for future works.

4.2.3 *Consultation - RL Policy.*

We use a two-layer feed forward neural network as our policy network. For the ease of convergence, we use the standard Deep Q-learning [18] for optimization[3]

[2] Unlike EAR, CPR does not specifically answer the questions of *how to adapt user feedbacks* like EARS by designing a reflection mechanism. It is because [13] reports it is still an open and challenging question with lots of details to be explored. Hence we leave it for future works.

[3] According to our experiments, Deep Q-learning does not lead to better model, while making the model much easier to converge. We also call the *value network* (the terminology in Deep Q-learning) as policy network for ease of discussion.

The policy network takes the state vector $\mathbf{s}$ as input and outputs the values $Q(\mathbf{s}, a)$ for the two actions, indicating the estimated reward for a_{ask} or a_{rec}. A system will always choose the action with higher estimated reward. The state vector $\mathbf{s}$ is a concatenation of two vectors:

$$\mathbf{s} = \mathbf{s}_{his} \oplus \mathbf{s}_{len}, \tag{5}$$

where $\mathbf{s}_{his}$ encodes the conversation history, which is expected to guide the system to act smarter, e.g., if the asked attributes are accepted for multiple turns, it might be a suitable timing to recommend. The $\mathbf{s}_{len}$ encodes the size of candidate item set. As discussed by [13], it is easier to make successful recommendations when there are fewer candidate items.

The reward follows [13], containing five kinds of rewards, namely, (1) r_{rec_suc}, a strongly positive reward when the recommendation succeeds, (2) r_{rec_fail}, a strongly negative reward when the recommendation fails, (3) r_{ask_suc}, a slightly positive reward when the user accepts an asked attribute, (4) r_{ask_fail}, a negative reward when the user rejects an asked attribute, and (5) r_{quit}, a strongly negative reward if the session reaches the maximum number of turns. The accumulated reward is the weighted sum of these five. The detailed value for each reward can be found in Sec 5.2.

While some components of our SCPR are adapted from EAR, it is worth highlighting two significant differences between them. First, SCPR leverages on the *adjacent attribute* constraint on the graph, largely reducing the search space of attributes. Second, SCPR scores attributes through message propagation on the graph, instead of by the policy network as what has been done in EAR. This enables our policy network to have a *much smaller decision space* — only two actions, alleviating the pressure for policy making.

5 EXPERIMENTS

In this section, we are going to evaluate our proposed CPR framework by empirically examining the SCPR implementation on two real-world datasets. We use the following research questions (RQs) to guide our experiment[4].

- **RQ1.** How does our CPR framework compared with existing conversational recommendation methods?
- **RQ2.** Are the *adjacent attribute constraint* and *smaller decision space* in SCPR really effective?
- **RQ3.** Can our method make the reasoning path explainable and easy-to-interpret?

5.1 Dataset Description

For better comparison, we follow EAR [13] to conduct experiments on LastFM[5] for music artist recommendation and Yelp[6] for business recommendation. LastFM contains 1,801 users and 27,675 items and 76,693 interactions. Yelp contains 27,675 users, 70,311 items and 1,368,606 interactions.

In the original paper of EAR [13], LastFM is designed to evaluate binary question scenario, where the user give preference towards an attribute using yes or no. For the ease of modeling, Lei et al. [13] manually merged relevant attributes into 33 coarse-grained attributes. Whereas, the Yelp dataset is designed for enumerated questions, where the user can select multiple attributes under one category. They manually built a 2-layer taxonomy and there are 29 first-layer categories with 590 second-layer attributes.

While we follow the setting of [13], we believe they are not necessarily the best practice as they requires heavy manual efforts with expert knowledge, which is expensive for real usage. Therefore we also consider the setting of using the original attributes (pruning off frequency < 10 attributes), denoting them as LastFM* (containing 8438 attributes) and Yelp* (containing 590 attributes) separately. The statistics can be found in Appendix A.

5.2 Experimental Setup

5.2.1 Training Details. We split each dataset for training, validation and testing in a ratio of 7:1.5:1.5. And set top k item as 10, and maximum turn T as 15. Following [13, 24] The training process is made up of two parts: (1) An offline training for scoring function of item in *reasoning* step. We use the historical clicking record in the training set to optimize our factorization machine offline (Eq. (3)) by strictly follow [13]. The goal is to assign higher score to the clicked item for each users. We articulate the details in Appendix B and we also refer the readers to the original paper [13] for more Information. All hyperparameters for offline training remains the same as [13]. (2) An online training for reinforcement learning used in consultation step. We use a user simulator (*c.f.* Sec 5.2.2) to interact with the user to train the policy network using the validation set. The detailed rewards to train the policy network are: r_{rec_suc}=1, r_{rec_fail}=-0.1, r_{ask_suc}=0.01, r_{ask_fail}=-0.1, r_{quit}=-0.3. The parameters of the DQN are empirically set as following: the experience replay memory size is 50,000, the sample batch size is 128, discount factor γ is set to be 0.999. We optimize policy network with RMSprop optimizer and update the target network every 20 epsiodes. It is also noticeable that, since our policy network have much smaller actions space and we adopt Deep Q-learning, the network is easier to converge. We do not need to have pre-training as adopted in EAR [13] and CRM [24]. All those hyperparameters related online training are tuned according to the validation set.

5.2.2 User Simulator For MCR. As a CRS is an interactive system, it needs to be trained and evaluated by interacting with users. However, it is infeasible to do so in a research lab. Employing a user simulator is a common practice [2]. We follow the user simulators in [13, 24] which simulate one conversation session for one user-item (u, v) interaction record in validation set (for training) and testing set (for testing). In a giving session, the user u's preference is anchored by item v: (1) when the system proposes a list of items, he will only accept it if the list contains item v; (2) when a system asks for an attribute, he will only confirm he likes it if this attribute is included by item v. There is no denying that such simulation has many limitation, but it is the most practical and realistic at current stage [13, 24]. One major attack for such simulation is that the user may "falsely" reject an item which is actually liked by him but it has not been observed hence not being clicked by him. However, it is hard to address it as there is few suitable exposure data. One may also suggest to treat all user-item interaction in testing set as positive instances for one session, we also forgo using it because the aim for CRS is to capture user's current specific preference

[4] Code and datasets can be found at: https://cpr-conv-rec.github.io/
[5] https://grouplens.org/datasets/hetrec-2011/
[6] https://www.yelp.com/dataset/

which may shift from his general interest. As our main focus is the strategy of graph reasoning, we use template for conversations.

5.2.3 Baselines. Although there are more CRS models, they follow different settings, hence being not comparable to us. We use the following baselines to compare:

- **Max Entropy**. This method follows a rule-based protocol to ask and recommend. When asking question, it always chooses an attribute with the maximum entropy within the current candidate item set. The system follows certain probabilities to recommend. Details can be found at [13].
- **Abs Greedy** [7]. This method serves as a baseline where the model only recommends items and updated itself, until it finally makes successful recommendation. Christakopoulou et al. [7] report that it outperforms online recommendation methods like Thompson Sampling [3].
- **CRM** [24]. This is a CRS model which records user's preference into a belief tracker, and uses reinforcement learning (RL) to find the policy to interact with the user. The RL leverages on a policy network whose state vector is the result of belief tracker. We follow [13] to adapt it to the MCR scenario.
- **EAR** [13]. This is the state-of-the-art method on MCR setting and proposed a three stage solution called Estimation–Action–Reflection which emphasizes on the interaction between conversation component and recommendation component. This inspires our SCPR implementation hence being the most comparable model.

5.2.4 Evaluation Metrics. The evaluation follows [13]. We use success rate (SR@t) [24] to measure the cumulative ratio of successful recommendation by turn t. We also use average turns (AT) to record the average number of turns for all session (if a session still fails in the last turn T, we count the turn for that session as T). Therefore, the higher SR@t indicates a higher performance at a specific turn t, while the lower AT means an overall higher efficiency.

5.3 Performance Comparison of SCPR with Exsiting Models (RQ1)

Table 2: Success Rate @ 15 and Average Turn. Bold number represents the improvement of SCPR over existing models is statistically significant ($p < 0.01$) (RQ1)

	LastFM		Yelp	
	SR@15	**AT**	**SR@15**	**AT**
Abs Greedy	0.222	13.48	0.264	12.57
Max Entropy	0.283	13.91	0.921	6.59
CRM	0.325	13.75	0.923	6.25
EAR	0.429	12.88	0.967	5.74
SCPR	**0.465**	**12.86**	**0.973**	**5.67**

Table 2 and 3 present the statistics of model's performances. We can see that our SCPR model achieves significantly higher SR and less AT than state-of-the-art baselines, demonstrating our SCPR method's supurior performances in usage.

Table 3: Performance comparison on original attributes. Bold number represents the improvement of SCPR over existing models is statistically significant ($p < 0.01$) (RQ1)

	LastFM*		Yelp*	
	SR@15	**AT**	**SR@15**	**AT**
Abs Greedy	0.635	8.66	0.189	13.43
Max Entropy	0.669	9.33	0.398	13.42
CRM	0.580	10.79	0.177	13.69
EAR	0.595	10.51	0.182	13.63
SCPR	**0.709**	**8.43**	**0.489**	**12.62**

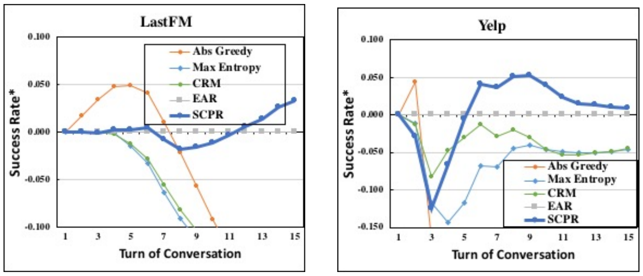


Figure 3: Success Rate* of compared methods at different turns on LastFM and Yelp (RQ1).

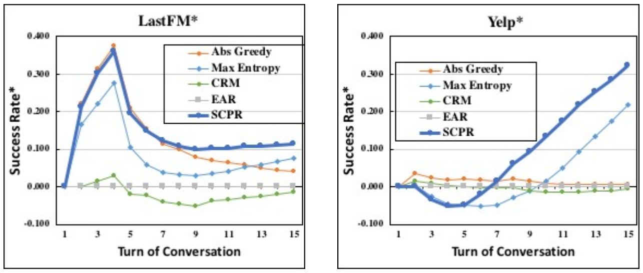
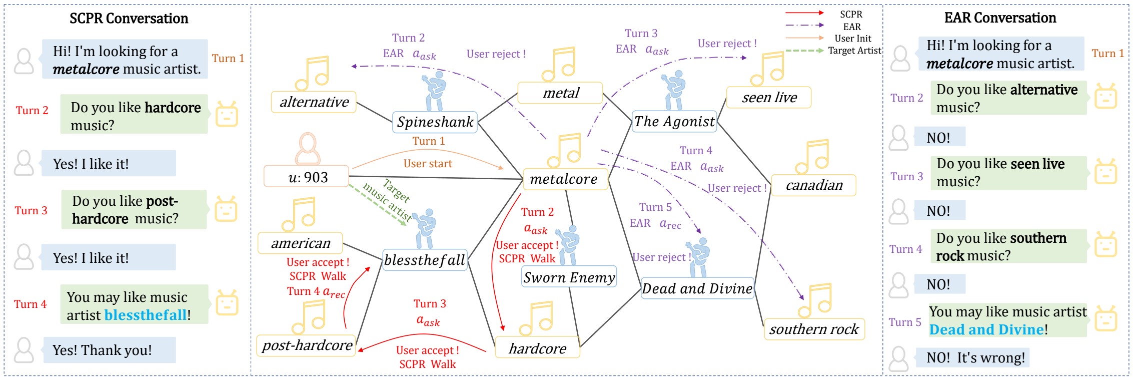


Figure 4: Success Rate* of compared methods at different conversation turns on LastFM* and Yelp*(RQ1).

We also intuitively present the performance comparison in Figure 3 and 4. They show Success Rate * (SR*) in each turn. SR* denotes the relative SR compared with the most competitive baseline EAR (meaning the difference of SR between each method and EAR), and EAR serves as the gray line of y = 0 in the figures. We have following discoveries:

- It is important to see our SCPR outperforms all baselines on various settings. Interestingly, we can find that our SCPR shows larger advantage in LastFM* and Yelp* datasets, showing its validity in practical usage with large attribute space. This also validates our key design in CPR. Firstly, the graph constraint helps our model to eliminate many irrelevant attributes to ask, this becomes especially helpful when there are a large number of attributes. In contrast, we discover that EAR has more difficulties in asking accurate attributes in LastFM* and Yelp*. Secondly, our framework utilizes a more dedicated RL model which only decides to recommend or to ask, hence have better chances to learn more effective policy. At the same time, EAR may outperform SCPR on first few rounds, but it falls behind in future rounds. It is due to EAR has much larger action space, making it challenging for them to learn effective strategy to recommend while asking.
- Interestingly, Abs Greedy can achieve the best results on the first few turns but plunges in further turns. The reason is that Abs Greedy is the only method that solely attempts to recommend



SCPR Conversation

Hi! I'm looking for a *metalcore* music artist. Turn 1

Turn 2 Do you like **hardcore** music?

Yes! I like it!

Turn 3 Do you like **post-hardcore** music?

Yes! I like it!

Turn 4 You may like music artist **blessthefall**!

Yes! Thank you!

EAR Conversation

Hi! I'm looking for a *metalcore* music artist. Turn 1

Turn 2 Do you like **alternative** music?

NO!

Turn 3 Do you like **seen live** music?

NO!

Turn 4 Do you like **southern rock** music?

NO!

Turn 5 You may like music artist **Dead and Divine**!

NO! It's wrong!

Figure 5: Sample conversations generated by SCPR (left) and EAR (right) and their illustrations on the graph (middle).

items to user. While Abs Greedy is continuously pushing recommendation, other methods are probably consulting user's explicit feedback on attributes which in turns helps reduce candidate item space and help the model achieve long term reward. This also validates our core design in SCPR – utilizing user's explicit feedback on attributes.

- The two previously proposed RL-based methods, EAR and CRM can both outperform max entropy in Yelp and LastFM, but achieve lower performance than max entropy in Yelp* and LastFM*. The reason is that Yelp* and LastFM* have larger attribute space (590 and 8438 than 29 and 33). According to their model design, their RL model is responsible for both *which attribute to ask* and *whether to recommend*. Therefore, they have action spaces of 590+1 dimensions and 8438+1 dimensions for Yelp* and LastFM* respectively. Such larger action space may bring challenges to action making.

5.4 Evaluating Key Design in SCPR (RQ2)

The key design of our SCPR method is that we leverage on the *adjacent attribute constraint* of the graph and a more dedicated RL model with smaller action space. To test the effectiveness of such key features, we conduct additional experiments by designing a variant of our SCPR model, named SCPR-v. Specifically, we replace our policy network with the policy network in EAR. It has the same state vector as EAR and the action space of the policy network increases from 2 to $|\mathcal{P}| + 1$, meaning that the policy function is also responsible for *deciding which attribute to ask*. Note that we keep other components unchanged, including our graph constraint of adjacent attributes. Such constraint exerts influence on our policy function in a straightforward way: we add a condition of "being one adjacent attribute" to the selection of action with maximum value. Therefore such SCPR-v model can be seen as an intermediate layer between EAR and SCPR, (1) it can be seen as a variant of SCPR where its RL model is not so dedicated, (2) it can also be seen as a variant of EAR where the attribute asking can be helped (if any) by our graph constraint. We follow the same implementation paradigm for SCPR-v. Due to the space limitation, we only briefly report the Success Rate* comparison among SCPR-v, EAR and SCPR on LastFM* and Yelp* datasets which are more representative.

We have these discoveries: (1) SCPR-v has generally worse performance than SCPR, since it is very challenging for decision making in a very large action space. It validates our design in a more dedicated RL model with small action space. Interestingly we can see that SCPR-v has similar performance at first few turns compared with SCPR, but falls behind in future turns. According to instance-level studies, we observe the RL component of SCPR-v adopts simple strategies. It asks a few attributes and recommend items at earlier turns than SCPR. Then with fewer attributes known, it has a larger candidate item space, making it harder to achieve higher SR in longer turns. It suggests that SCPR-v with very large action space has difficulty learning effective strategy for long term reward. (2) We can see that SCPR-v has better performance than EAR. This advantage is due to our graph constraint on attributes that eliminates many irrelevant attributes, making it easier for attribute choice.

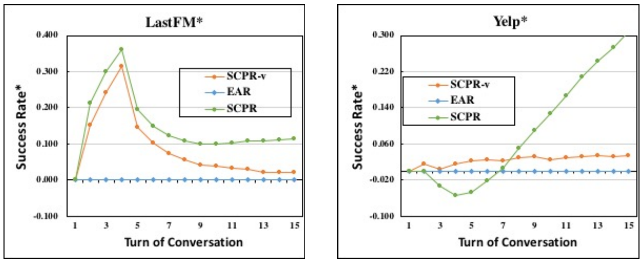


Figure 6: Success Rate* of compared methods at different conversation turns on LastFM* and Yelp*(RQ2).

5.5 Case Study on Explainability (RQ3)

Aside from the superior performance on success rate and average turns, our CPR is also more explainable. It conducts conversational recommendation by walking (reasoning) on graph, resulting in a path of attributes. The path brings crystally clear reasoning logic which is naturally explainable.

Let's go through an example of real interaction from LastFM* in Figure 5. We display the conversation histories of SCPR and EAR on two sides of the figure, and illustrate the whole processing in the middle graph. The session is initiated by the user (id: 903) who specifies an attribute he likes as "*metalcore*". We can see our SCPR travels a short path of attributes ("*metalcore*" to "*hardcore*" then to "*post-hardcore*") that quickly reaches user's preferred item (artist "*blessthefall*") and successfully make recommendation. The whole conversation is coherent and the red path is the explanation of the recommendation reason. On the contrary, EAR's behavior looks strange. It first asks "*alternative*", then asks "*seenlive*" followed by "*southern rock*". Those attributes are not very closely related, being more like a random pop-ups of attributes. From model developing perspective, such loss of relevance makes it hard to explain why the EAR executes such jumps. From application perspective, such loss of relevance leads to less coherent conversations.

6 CONCLUSION AND FUTURE WORK

We are the first to introduce graph to address the multi-round conversational recommendation problem, and propose the *Conversational Path Reasoning* (CPR) framework. CPR synchronizes conversation with the graph-based path reasoning, making the utilization of attribute more explicitly hence greatly improving explainability for conversational recommendation. Specifically, it tackles *what item to recommend* and *what attribute to ask* problems through message propagation on the graph, leveraging on the complex interaction between attributes and items in the graph to better rank items and attributes. Using the graph structure, a CRS only transits to the adjacent attribute, reducing the attribute candidate space and also improving the coherence of the conversation. Also, since the items and attributes have been ranked during the message propagation, the policy network only needs to decide *when to ask and when to recommend*, reducing the action space to be 2. It relieves the modeling load of the policy network, enabling it to be more robust especially when the candidate space is large.

There are many interesting problems to be explored for CPR. First, CPR framework itself can be further improved. For example, CPR does not consider how to adapt the model when the user rejects a recommended item. How to effectively consider such rejected items would be an interesting and challenging task. Second, more sophisticated implementation can be considered. For example, it is possible to build more expressive models for attribute scoring other than the weighted max-entropy as adopted in this paper. Currently, the embeddings of items and attributes do not get updated during the interactive training. It would be better to build a more holistic model to incorporate the user feedback to update all parameters in the model, inclusive of user, item and attribute embeddings.

Acknowledgement: This research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative as well as National Natural Science Foundation of China (61972372, U19A2079). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors. We thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] Keping Bi, Qingyao Ai, Yongfeng Zhang, and W Bruce Croft. 2019. Conversational product search based on negative feedback. In *CIKM*. 359–368.

[2] Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefevre, and Olivier Pietquin. 2011. User simulation in dialogue systems using inverse reinforcement learning. In *Interspeech 2011*. 1025–1028.

[3] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *NeurIPS*. 2249–2257.

[4] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale interactive recommendation with tree-structured policy gradient. In *AAAI*, Vol. 33. 3312–3320.

[5] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. In *EMNLP-IJCNLP*. 1803–1813.

[6] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. 2018. Q&R: A Two-Stage Approach toward Interactive Recommendation. In *SIGKDD*. 139–148.

[7] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *SIGKDD*. 815–824.

[8] Sudeep Gandhe and David Traum. 2008. An evaluation understudy for dialogue coherence models. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*. 172–181.

[9] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*. 355–364.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[11] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. 549–558.

[12] Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Zhao, and Dawei Yin. 2018. Explicit State Tracking with Semi-Supervisionfor Neural Dialogue Generation. In *CIKM*. 1403–1412.

[13] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation–Action–Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *WSDM*. 304–312.

[14] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures. In *ACL*. 1437–1447.

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

[16] Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2020. Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-Start Users. *arXiv preprint arXiv:2005.12979* (2020).

[17] Lizi Liao, Yunshan Ma, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2018. Knowledge-aware Multimodal Dialogue Systems. In *ACM MM*. 801–809.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[19] Bilih Priyogi. 2019. Preference Elicitation Strategy for Conversational Recommender System. In *WSDM*. 824–825.

[20] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.

[21] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[22] Nicola Sardella, Claudio Biancalana, Alessandro Micarelli, and Giuseppe Sansonetti. 2019. An Approach to Conversational Recommendation of Restaurants. In *ICHCI*. 123–130.

[23] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *WSDM*. 555–563.

[24] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *SIGIR*. 235–244.

[25] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[26] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*, Vol. 33. 5329–5336.

[27] Ji Wu, Miao Li, and Chin-Hui Lee. 2015. A probabilistic framework for representing dialog systems and entropy-based dialog management through dynamic stochastic state evolution. *TASLP* 23, 11 (2015), 2026–2035.

[28] Qingyun Wu, Naveen Iyer, and Hongning Wang. 2018. Learning Contextual Bandits in a Non-stationary Environment. In *SIGIR*. 495–504.

[29] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*. 285–294.

[30] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *RecSys*. 140–144.

[31] Tong Yu, Yilin Shen, and Hongxia Jin. 2019. An Visual Dialog Augmented Interactive Recommender System. In *SIGKDD*. 157–165.

[32] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, and Changyou Chen. 2019. Text-Based Interactive Recommendation via Constraint-Augmented Reinforcement Learning. In *NIPS*. 15188–15198.

[33] Xiaoying Zhang, Hong Xie, Hang Li, and John Lui. 2020. Conversational Contextual Bandit: Algorithm and Application. In *WWW*.

[34] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *CIKM*. 177–186.

[35] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral Collaborative Filtering. In *RecSys*. 311–319.

[36] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *SIGIR*. 315–324.

A DATASET STATISTICS

Table 4: Dataset Statictics of LastFM and Yelp. Here we list the relation types in different datasets to let readers to get better understanding of the dataset.

Dateset		LastFM	Yelp
User-Item Interaction	#Users	1,801	27,675
	#Items	7,432	70,311
	#Interactions	76,693	1,368,606
	#attributes	**33**	**29**
Graph	#Entities	9,266	98,605
	#Relations	4	3
	#Triplets	138,217	2,884,567
Relations	**Description**	**Number of Relations**	
Interact	user $\Longleftrightarrow$ *item*	76,696	1,368,606
Friend	user $\Longleftrightarrow$ *user*	23,958	688,209
Like	user $\Longleftrightarrow$ *attribute*	7,276	*
Belong_to	item $\Longleftrightarrow$ *attribute*	30,290	350,175

Table 5: Dataset Statistics for LastFM* and Yelp*, we use original attributes to avoid complex feature engineering.

Dateset		LastFM*	Yelp*
User-Item Interaction	#Users	1,801	27,675
	#Items	7,432	70,311
	#Interactions	76,693	1,368,606
	#attributes	**8,438**	**590**
Graph	#Entities	17,671	98,576
	#Relations	4	3
	#Triplets	228,217	2,533,827
Relations	**Description**	**Number of Relations**	
Interact	user $\Longleftrightarrow$ *item*	76,696	1,368,606
Friend	user $\Longleftrightarrow$ *user*	23,958	688,209
Like	user $\Longleftrightarrow$ *attribute*	33,120	*
Belong_to	item $\Longleftrightarrow$ *attribute*	94,446	477,012

B DETAILS OF OFFLINE TRAINING IN THE *REASONING* STEP

In our **CPR** framework design, there is a trainable component in *reasoning* step for item scoring. For simplicity, we instantiate it as the FM model in EAR[13]. For the reproducibility of this paper, we articulate the whole process of such instantiation in this section.

B.1 Training Objective

EAR[13] embeds users, items and attributes as vectors into one Factorization Machine (FM)[21] model. The training objective for such FM model is simultaneously achieving item prediction and attribute prediction for Multi-round Conversational Recommendation(MCR) scenario, using a multi-task pairwise loss.

B.1.1 Item Prediction.

We borrow a variant of Factorization Model (FM) model as introduced in [13] to capture the interaction between users, items and attributes. As discussed in Section 4.1, the scoring function is defined as:

$$f(u, v, \mathcal{P}_u) = \mathbf{u}^T \mathbf{v} + \sum_{p_i \in \mathcal{P}_u} \mathbf{v}^T \mathbf{p_i}, \tag{6}$$

where the first and second term represent message propagated from user to item and item to user respectively.

We follow [13] to use a pairwise loss to optimize, one key innovation is that they use two types of negative samples $\mathcal{D}_1$ and $\mathcal{D}_2$ tailored for MCR:

$$\begin{aligned} L_{item} = & \sum_{(u,v,v') \in \mathcal{D}_1} -\ln\sigma(f(u,v,\mathcal{P}_u) - f(u,v',\mathcal{P}_u)) \\ & + \sum_{(u,v,v') \in \mathcal{D}_2} -\ln\sigma(f(u,v,\mathcal{P}_u) - f(u,v',\mathcal{P}_u)) + \lambda_\Theta \, \|\Theta\|^2 , \end{aligned} \tag{7}$$

where

$$\mathcal{D}_1 := \{(u, v, v') \mid v' \in \mathcal{V}_u^-\}, \quad \mathcal{V}_u^- := \mathcal{V} \backslash \mathcal{V}_u^+ \tag{8}$$

$$\mathcal{D}_2 := \{(u, v, v') \mid v' \in \widehat{\mathcal{V}_u^-}\}, \quad \widehat{\mathcal{V}_u^-} := \mathcal{V}_{cand} \backslash \mathcal{V}_u^+ \tag{9}$$

The intuition is that the model first needs to learn user's general preference ($\mathcal{D}_1$). Additionally it also should learn user's preference when some attributes have been confirmed, resulting a dynamically updating candidate item set $\mathcal{V}_{cand}$, which is a main characteristic for MCR ($\mathcal{D}_2$). Specifically, $\mathcal{V}_u^-$ is the ordinary negative samples which are non-interacted items, and $\widehat{\mathcal{V}_u^-}$ is candidate item set $\mathcal{V}_{cand}$ that excludes the interacted items $\mathcal{V}_u^+$. The obtaining of $\mathcal{V}_{cand}$ is a dynamic process, which will be discussed in Section B.2.

B.1.2 Attribute Prediction.

The EAR[13] also leverages on the FM model to make attribute prediction. Intuitively, the next attribute p to ask should be dependent on the confirmed attribute set $\mathcal{P}_u$, formally:

$$\hat{g}(p|u, \mathcal{P}_u) = \mathbf{u}^T \mathbf{p} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}^T \mathbf{p}_i, \tag{10}$$

where the first term captures user's general preference towards the given attribute p, and the second term models the interaction between p and each attribute in the confirmed attribute set $\mathcal{P}_u$.

They similarly leverages on the pairwise loss for attribute prediction:

$$L_{attr} = \sum_{(u,p,p') \in \mathcal{D}_3} -\ln\sigma(\hat{g}(p|u,\mathcal{P}_u) - \hat{g}(p'|u,\mathcal{P}_u)) + \lambda_\Theta \|\Theta\|^2 , \tag{11}$$

where the pairwise training data $\mathcal{D}_3$ is defined as:

$$\mathcal{D}_3 = \{(u,p,p')|p \in \mathcal{P}_v, p' \in \mathcal{P} \backslash \mathcal{P}_v\}, \tag{12}$$

The $\mathcal{P}_v$ here denotes attributes of item v, hence p and p' represent attributes that belongs and *not* belongs to current item respectively, forming the pairwise sample.

B.1.3 Multi-task learning.

Since [13] discovered that item prediction and attribute prediction can mutually promote, we also follow their practice to use such multi-task pairwise loss to achieve these two goals:

$$L = L_{item} + L_{attr}. \tag{13}$$

B.2 Data Collection

As we have elaborated the training objective of the FM model used in the *reasoning* step, now we are going to describe how we obtain the data used to train such model, which are in fact $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$.

As a common practice introduced in Section 5.2.2, we also leverage on user simulator to obtain such data. As introduced before, we use observed user-item interactions to ground such simulation. We accumulate $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ through many MCR sessions and append new instances at each steps of the interactions. Specifically, given a user u and an item v which has an attribute set $\mathcal{P}_v$, without the loss of generality, we assume $\mathcal{P}_v = \{p_0, p_1, p_2, p_3, p_4\}$. As for the accumulation of $\mathcal{D}_1$, it is actually independent from the interaction steps because it is intrinsically static. Therefore we directly sample one item from the non-interacted items of user u. Now let's assume we are at the stage where user has confirmed a few attributes, yielding the confirmed attribute set $\mathcal{P}_u = \{p_0, p_1, p_2\}$. Now $\mathcal{V}_{cand}$ is the set of items satisfying all attributes in $\mathcal{P}_u$, one negative instance will be sampled from the non-interacted items in $\mathcal{V}_{cand}$ to form $\mathcal{D}_2$. Note that the positive instances in pairwise samples are always v for both $\mathcal{D}_1$ and $\mathcal{D}_2$. Finally, as for the attribute side, the positive instances for attributes are $\{p_3, p_4\}$, each of them will be paired with a negative instance sampled from $\mathcal{P} \backslash \mathcal{P}_v$ and be added into $\mathcal{D}_3$.

In order to have a high coverage of the dataset, we use all user-item interactions in training set to ground such simulation. What's more, we simulate multiple times for each user-item interaction, with all possibility of the first attribute user informed p_0 being tried.

B.3 Training Details

After the training data has been collected, we strictly follow the training instruction in [13]. To briefly report, we set the embedding size of FM model as 64. We used SGD optimizer with L2 regularization of 0.001. The learning rate for item prediction task and attribute prediction task are set us 0.01 and 0.001 repectively.