Improving Conversational Recommendation System Through Personalized Preference Modeling and Knowledge Graph

Feng Wu, Guoshuai Zhao, *Member, IEEE,* Tengjiao Li, Jialie Shen, Xueming Qian, *Member, IEEE,*

Abstract—Conversational recommendation systems (CRS) can actively discover users' preferences and perform recommendations during conversations. The majority of works on CRS tend to focus on a single conversation and dig it using knowledge graphs, language models, etc. However, they often overlook the abundant and rich preference information that exists in the user's historical conversations. Meanwhile, end-to-end generation of recommendation results may lead to a decrease in recommendation quality. In this work, we propose a personalized conversational recommendation system infused with historical interaction information. This framework leverages users' preferences extracted from their historical conversations and integrates them with the users' preferences in current conversations. We find that this contributes to higher accuracy in recommendations and fewer recommendation turns. Moreover, we improve the interactive pattern between the recommendation module and the dialogue generation module by utilizing the slot filling method. This enables the results inferred by the recommendation module to be integrated into the conversation naturally and accurately. Our experiments on the benchmark dataset demonstrate that our model significantly outperforms the state-of-the-art methods in the evaluation of recommendations and dialogue generation.

Keywords—Conversational Recommendation System, Dialogue Generation, Personalized Recommendations

1 Introduction

With the increasing cost of e-commerce platforms, conversational recommendation system (CRS) has become a hotspot among researchers. Conversational recommendation systems are designed to chat with users and provide personalized recommendations (e.g., movies, restaurants, cosmetics and commodities) [1]. Different from traditional recommendation systems which passively gather information from users, CRS uses natural language to interact with users to understand user needs and provide suitable items to address current user needs [2]. On the other hand, CRS is also a kind of task-oriented dialogue generation system, and the purpose is to use as few dialogue rounds as possible to help users complete pre-determined tasks or actions [3]. Thus, CRS is regarded as a combination of a recommendation system and a dialogue system.

Traditional CRS consists of two parts: the recommendation module and the dialogue generation module. The recommendation module needs to understand users' preferences through multi-round dialogue context and reason out suitable items. Due to the fact that expressions provided by users often contain insufficient and implicit information, external knowledge has been introduced into the system. For example, existing studies [4]–[7] utilize knowledge graphs and review information to improve recommender components, providing precise recommendations. On the other hand, the conversation generation module is also an area for improvement to enhance the performance of the system. Recent works [8] applied Large-Scale Pre-trained Language Models (PLMs) such as BERT [9], Dialog-GPT [10], and BART [11], or presented new pre-trained models for CRSs to generate fluent and diverse dialogue responses.

However, these existing studies on CRS suffer from two major issues. First, previous CRSs require multiple rounds of dialogue to collect sufficient entities mentioned to search for relevant items for users. However, in multiple rounds of dialogue, users will inevitably become bored with constantly receiving incorrect information or frequently being asked questions. That is, although users hope to use as few rounds as possible to get a desirable recommendation, the system cannot make inferences with insufficient information [3]. Practically speaking, users who interact with the system multiple times are more sensitive to the number of conversation rounds, and these users constitute the main part of the system's user base. Ideally, a CRS should accumulate

- *F. Wu, G. Zhao, T Li are with the School of Software Engineering, Xi'an Jiaotong University, 710049, Xi'an, China. E-mails: E-mail: wufeng@stu.xjtu.edu.cn; guoshuai.zhao@xjtu.edu.cn; litengjiao58137@stu.xjtu.edu.cn*
- *X. Qian is with the Ministry of Education Key Laboratory for Intelligent Networks and Network Security and with SMILES LAB, Xi'an Jiaotong University, 710049, Xi'an, China. E-mail: qianxm@mail.xjtu.edu.cn*
- *J. Shen is with Department of Computer Science, City, University of London, London, UK. E-mail: jerry.shen@city.ac.uk (Corresponding author: Guoshuai Zhao.)*

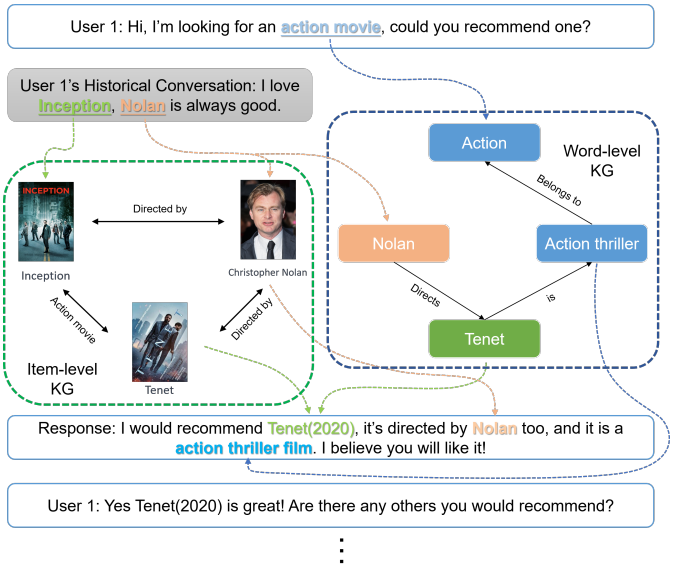

Fig. 1: An example of conversational recommendation with historical interaction data. When using conversation history information, the system can give recommendations without asking for further information.

users' historical information on its recommendations to reduce the rounds of dialogue when a user interacts with the system for a second time. Secondly, recent CRSs [4]–[6] employ a general encoder-decoder framework and suffer from overemphasizing the information from the current text. Due to the decoder usually receiving features from words and items, it is prone to overfitting on this information. In other words, if the format of sentences is similar, the system tends to generate the same answers and does not incorporate the results of the recommendation module. This causes a loss of result transformation from the recommendation module to the generation module. Finally, existing models rarely consider the impact of user interest shifts on model performance [12]. However, users' focus is unpredictable and difficult to model; they often open a new topic during a dialogue process and require CRSs to give recommendations that are irrelevant to the previous information. Before the emergence of generation-based CRS, traditional Question-based models [13], [14] struggled to grasp this point. Their paradigm of "system asks, user answers" found it challenging to provide accurate recommendations in response to shifts in user interests.

In this paper, we propose a Personalized Conversational recommendation system infused with Historical Interaction information, i.e., PCHI. PCHI consists of two major components: a history-enhanced recommender and a transformer-based generator. The recommender employs a KG-enhanced neural network for training, incorporating entity information extracted from the user's history of conversations. As shown in Fig. 1, the user's historical conversations can serve as an external knowledge source to model the user's preferences in collaboration with the knowledge graph. Using the information from the user's historical conversations, the recommendation module can capture the user's interests in the early stages of the conversation and provide accurate recommendations. The generator will generate a response with item slot, through multi-head self-attention layers, instead of combining context and recommendation data to generate the full conversation. After training, the item selected by the recommender will be filled into the reserved slot to perform recommendation and response generation. For example, If the system decide recommend the movie "Spider-Man", then the dialogue module's output should be: "I recommend [item]", where the item will be filled in with the item deemed most likely by the recommendation part during the module combination, finally resulting in the complete output of "I recommend Spider-Man." The results show that our model can provide high-quality and fewer-rounds performance compared to other models that introduce external knowledge.

The contributions of this work are summarized as follows:

1) We propose a method to combine the user's historical interaction information into the traditional conversational recommendation system, which contributes to reducing the conversation turns and achieving a higher level of recommendation hits.
2) We propose a novel fusion method that can smoothly and accurately incorporate recommendation results into dialogue generation tasks. It can improve the gap between recommendation and generation results in existing models.
3) Experimental results on the standard dataset show that the proposed method significantly outperforms state-of-the-art CRS methods in both recommendation and generation. The qualitative results also demonstrate that the proposed method can handle user interest shifts and reduce the number of recommendation turns.

2 Related Work

In this section, we introduce the existing works on conversational recommendation systems. Conversational recommendation systems, as an extension of traditional recommender systems, help address many potential limitations. For example, in some situations, users' current needs are highly dependent on the context instead of their past interactions, or users construct their preferences during the interaction with the system as they realize the scope of the options. Generally speaking, the idea behind such systems is that they support task-oriented, multi-turn dialogues with their users. CRSs can be divided into two categories [15]: Question-based systems and Generation-based systems.

2.1 Question-Based system

The question-based CRSs [16]–[18] provide recommendations through multi-round clarifying questions to users and extracting preference information from users' feedback. By narrowing down the possible item space by asking item-related questions [19], the system can ultimately provide an optimal set of items that fit users' preferences. Although this question-based system can achieve better performance in recommendation, the freedom and flexibility of dialogues are limited by passive responses. Recent works employ reinforcement learning-based approaches, bandit-based approaches, and graph-based approaches [20]–[23]. Zhang et al. [18] design a unified framework for conversational search and recommendation and propose an architecture called Multi-Memory Network (MMM) to accomplish the framework. Zou et al. [24] propose a model based on matrix factorization and infer user preferences and beliefs over items using Generalized Binary Search. Wong et al. [7] construct a billion-scale conversation knowledge graph in an e-commerce environment and fuse user-state and dialogue-interaction representations for Click-Through Rate predictions. Some studies [13], [14], [25] have also noticed the influence of historical dialogues on dialogue recommendation tasks, for example, Zhou et al. [26] propose adopting historical information and attribute-based preferences into the pre-training process to enhance information fusion between items and attribute-based preferences. Different from Question-based works, our model not only focuses on the accuracy of recommendation items but also improves the generation module to generate fluent and reliable responses for answering. Furthermore, Zhou employs a negative sample generator for constructive learning, which is not available in our case. To reduce the number of interactions required to find a suitable item, Li et al. [27] take a ranking optimization approach based on latent linear critiquing for multi-step conversational recommendation.

2.2 Generation-based system

Compared to the Question-based system, the Generation-based system is a free-style and natural CRS, which dynamically obtains user preferences through interactive conversation with users. However, after acquiring the ability for natural conversation, Generation-based systems tend to see a decrease in recommendation efficiency due to their end-to-end generation characteristics. Therefore, a good system should incorporate items into responses with natural and accurate languages. Some previous studies [28]–[31] introduce a pre-trained language model and template based model on generation modules e.g. BERT [9], GPT-2 [32]. Yang et al. [31] directly convert the items to embeddings by PLM and incorporate the item embedding and dialogue context to reduce the complexity of the model. Meanwhile, some researchers focus on the combination between the recommendation module and generation module to get better performance on the end-to-end results. Liang [33] propose a method that generates a response template with slot tied target items and context to incorporate recommendation and generation. Zhang et al. [34] divide the recommendation process into three sub-goals: Question answering, Chitchat about movie and Movie recommendation, and propose a unified framework which predict the sub-goals in dialogue context and generate dialog response though a noisy knowledge filter. Zhu et al. [35] design a retrieval-based knowledge-grounded multi-task learning framework, where the final response are selected by the predicted knowledge and context. On the other hand, in order to understand user's preferences and enhance the quality of recommendation, systems introduce external knowledge bases e.g., knowledge graphs (KGs). Zhou et al. [5] introduce knowledge graphs into the CRS, which enhance the data representation by a word-oriented KG and an entity-oriented KG. Recent works have employed tree-structured reasoning [36], subgraph construction [37], new knowledge graph [38], [39] and review information [6] to enhance the performance of KG in recommendation modules. To utilize multi-type external data, Zhou et al. [40] design a coarse-to-fine contrastive learning framework to improve various data semantic fusion. However, previous work often overlooked the fact that users' historical interaction records contain a wealth of information.

Our model is more similar to the Generation-based system; however, previous studies rarely consider users' historical interaction information in CRSs. They often focus on extending external knowledge sources rather than incorporating data from historical interaction sequences. Incorporating historical data into users' preference representation is an accessible and low-complexity approach for the entire CRS. Our model utilizes users' interaction sequences to improve the performance of recommendations and usability in the final response. Moreover, inspired by recent studies, we enhance the fusion method between the recommendation process and dialogue generation to achieve better end-to-end recommendation results.

3 Our Model

In this section, we present the PCHI. As shown as Figure 2 , our model consists of four components: 1) a personalization encoding module, 2) a dialogue generation module, and 3) a recommendation module. We will introduce the details of each component in the rest of this section.

3.1 Preliminary

Generally, a t-turn conversation context is denoted as $C = (t_1, t_2, t_3 \dots t_n)$, where t represents the utterances of the dialogue history given by the user and system. At t-th turn, system should receive utterances given by user with vocabulary V , and recommendation module

will find the appropriate item i_t from the total item set I based on the information in user utterances or decide not to give the recommendation if the i_t is equal to $\emptyset$. After that, the generation module generates natural language sentences s_t combined with proper item i_t from recommendation module. When the recommendation module thinks there are no recommended items to recommend, the generation module is likely to ask questions or generate some chit-chat responses to further explore the user's interests.

3.2 Personalization Encoding Module

To improve the responsiveness and accuracy of the recommendation system, we employ the information from users' history dialogues. For a user who is interacting with the system, we extract the user's preferred items in the historical conversations and obtain the user historical preferences item set $U_i = i_1, i_2, i_3 \ldots, i_n$. We adopt Item2Vec [41] to embed the user preferences information in historical conversations to learn entity representations. Specifically, for a given item set U_i , the objective function is as following:

$$\frac{1}{K}\sum_{j=1}^{K}\sum_{j\neq i}^{K} logp\left(I_j|I_i\right), \tag{1}$$

where K is the length of sequence U_i. $p(I_j|I_i)$ is the selection probability of an item j within a large set of item candidates U_i, and $p\left(I_j|I_i\right)$ is the softmax function:

$$p\left(I_j|I_i\right) = \frac{exp(u_i^T v_j)}{\sum_k exp\left(u_i^T v_k\right)}, \tag{2}$$

where u_i and v_i are latent vectors that correspond to the target and context representations for item i_n. Specifically, u_i represents the representation of a certain item i that we need, which is the target representation, while v_j represents the representation of an item that appears in a user's access records and has a similar number of visits to item i, which we refer to as the context representation. To avoid the computational complexity of $p\left(I_j|I_i\right)$, the softmax function is always replaced by negative sampling:

$$p\left(I_j|I_i\right) = \sigma\left(u_i^T v_j\right)\prod_{k=1}^{N}\sigma\left(-u_i^T v_k\right), \tag{3}$$

where $\sigma\left(x\right) = \frac{1}{exp(-x)}$, N is a parameter of the number of negative examples for each positive sample.

Then, we train the Item2Vec using gradient descent, and get the final representation for all the items. For each dialogue, we can acquire the users' historical preference sequence with embedding items $h_{u1}, h_{u2}, h_{u3} \ldots h_{ut}$, where h_{ut} denotes the d-dimensions vectors of item i_t. To integrate the information from one user, we stack these vectors to get the representation $v_u = h_{u1}, h_{u2}, h_{u3} \ldots h_{ut}$.

It is not sufficient to rely solely on the user's historical preferences in the conversation recommendation task; the information mentioned in the conversation is also crucial. Inspired by previous work [4], [5], we also employ the Knowledge Graph from DBpedia and ConceptNet as external knowledge. To utilize the word information in context, we adopt the Graph Convolutional Neural network (GCN) [42] to encode the ConceptNet [43] KG. Due to the complex relationships in ConceptNet do not significantly aid word embeddings [5], we only utilized the node information from ConceptNet as input for our GCN model. For each n layer of GCN, the new d-dimensional node feature matrix ${h_c}^{(n)} \in R^{L\times d}$ is calculated as follows:

$${h_c}^{(n)} = ReLU\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}{h_c}^{n-1}W^n\right), \tag{4}$$

where W^n is a weight matrix at the n-th layer, $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix of graphs. ${h_c}^{n-1}$ represents the lower-order neighborhood information. After the training, we can obtain the representation n_w for a word w. On the other hand, for mentioned entities and related items in context, we extract them from the dialogues and generate the entity subgraph. Then, we adopt Relational Graph Convolutional Network (RGCN) [44] to embed the relational information in DBpedia [45] to learn relational representations. Specifically, the representation of an entity e at n-th layer $h_e^{(n)} \in R^k$ is computed as follow:

$$h_e^{(n)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,j}}W_r^{(n-1)}h_j^{(n-1)} + W_e^n h_e^{(n-1)}\right), \tag{5}$$

where N_i^r denotes the neighbor set of entity e under the relation R, W_r^{n-1} denotes a learnable relation-specific transformation matrix for the embeddings from the neighboring node at the $(n-1)$-th layer. W_e^n is a learnable matrix for transforming the representations of entity e at n-th layer. $c_{i,j}$ is a normalization factor. The representations construct a search space of recommended candidates for item retrieval.

At this point, we obtain the users' preference representation v_u from historical interaction data, word-oriented representation h_c and item-oriented representation h_e. To obtain the final user preference representation, we extract the entity representations used in the context and integrate them.

3.3 Recommendation Module

To integrate the external knowledge into dialogues, we extract the related entities (i.e. items and words) sequence $E_u = e_1, e_2, e_3 \ldots e_u$ in dialogue context C. Looking up the hidden representations of entities in E_u from h_c and h_e, we can get all entities' hidden representation $H_c = h_{e1}, h_{e2}, h_{e3} \ldots {}_{e=1}^{E_u}$ and $H_e = h_{e1}, h_{e2}, h_{e3} \ldots {}_{e=1}^{E_u}$in the dialogue. Usually, a conversation is made up of a

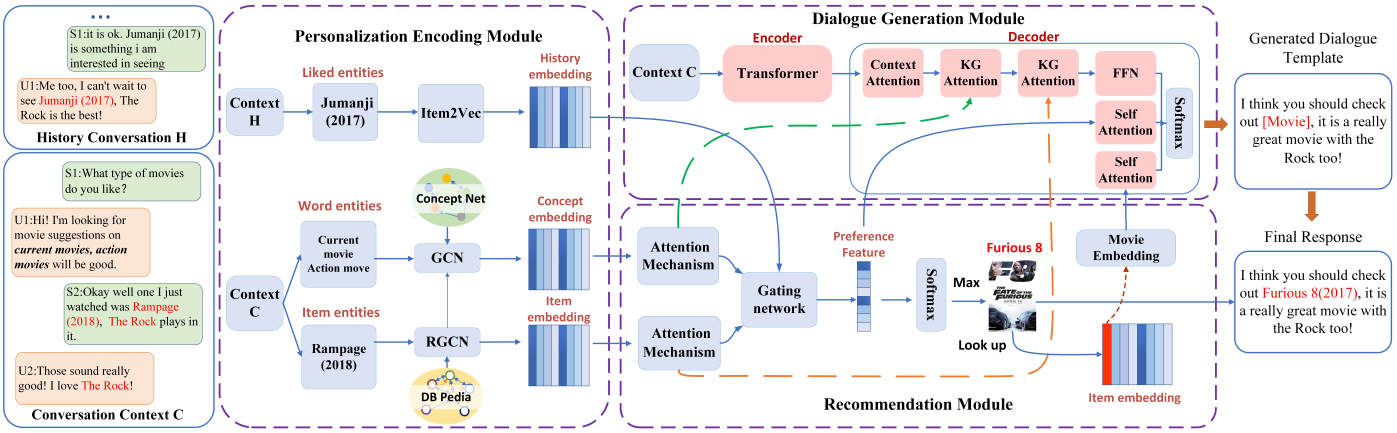

Fig. 2: The overall of proposed model, where context C and context H denote conversation context and history context, respectively.

number of contextual words and related items. Some of them are noise, which will affect the final performance. Therefore, we utilize the attention mechanism [46] on H to learn each entity's importance comprehensively. The distribution M_e vectors is calculated as follows:

$$M_e = aH \tag{6}$$

$$a = softmax\left(w_{s2} tanh\left(w_{s1} H^T\right)\right) \tag{7}$$

where w_{s2} and w_{s1} are two learnable weight matrix, a is importance vector for each entities. H is the contextual word representations or item set representations.

After attention mechanism, we have word vector v_c , item vector v_e, and users' historical preference vector v_u. We use the gate mechanism to integrate preference representation p_u for the user u.

$$p_u = \omega_1 v_c + \omega_2 v_e + \omega_3 v_u, \tag{8}$$

$$\omega_i = softmax(w_{gi} \cdot v_i), \tag{9}$$

where w_{gi} are learnable parameters, w_i are the weights of the word vectors, item vectors and user's historical preferences vectors, respectively. We employ the p_u to calculate the probability of the recommendation item for the user u:

$$P_{rec} = softmax\left(p_u H_i^{\top}\right), \tag{10}$$

where H_i is the hidden representation of item i.

3.4 Dialogue Generation Module

To generate the response, we choose a transformer-based model that has performed effectively on the dialogue generation task. Inspired by previous works [5], [33], we use the standard Transformer encoder architecture and the KG-enhanced decoder. Then, we mask all the items in the dialogue context with [ITEM] tokens, which means that the decoder will generate a response with words from the vocabulary and [ITEM] tokens instead of a complete response. Specifically, in the encoder stage, given the context C, we can calculate the embedding representation E_c:

$$E_c = Transformer\left(C\right), \tag{11}$$

upon encoding the context utterance, we enhance the KG-integrated decoder that is capable of generating more comprehensive responses by leveraging information from the KG, as well as combining user preference hidden representations provided by the recommendation module and item embeddings from the KG. Formally, we define E_d^n as the embedding matrix output from decoder n-th layer as follows:

$$A_0^n = MHA\left(E_d^{n-1}, E_d^{n-1}, E_d^{n-1}\right), \tag{12}$$

$$A_1^n = MHA\left(A_0^n, E_c, E_c\right), \tag{13}$$

$$A_2^n = MHA\left(A_1^n, v_c, v_c\right), \tag{14}$$

$$A_3^n = MHA\left(A_1^n, v_e, v_e\right), \tag{15}$$

$$E_d^n = FFN\left(A_3^n\right), \tag{16}$$

where E_c is the output of encoder, v_c and v_e are words vector and item vector generated by recommendation module, respectively. $MHA(Q, K, V)$ defines the multi-head attention function that takes a query matrix Q, a key matrix K, and a value matrix V as input and outputs the attentive value matrix:

$$MHA\left(Q, K, V\right) = Concat\left(head_1, \ldots, head_h\right) W^O, \tag{17}$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right), \tag{18}$$

then $FNN(\cdot)$ is a fully connected feed-forward network, which consists of two linear transformations with a $ReLU$ activation.

$$FFN\left(x\right) = max\left(0, xW_1 + b_1\right) W_2 + b_2, \tag{19}$$

now we have the output from the decoder. In previous work, this output was used directly to calculate the probability of the sequence, in order to generate usable sentences, which led to some problems. For instance, the decoder might generate a sentence containing item tags, but the recommender would not generate corresponding items because it could not capture enough information from the current context. To address this issue, we introduce the user's preference embeddings generated in Section 3.1, which effectively combine information from the generated response, dialogue context, and candidate item set. Specifically, we take the output of the decoder, and combine it with the user representation and item embedding to compute the probability distribution using an MLP layer:

$$P_{gen} = softmax(MLP([E_d, p_u, v_e^i]) \tag{20}$$

where P_{gen} is the probability of next token in generation sequence, $MLP(\cdot)$ means the multi-layer perception, E_d^n, p_u and v_e^i are decoder output, user preference representation and item embedding of the recommendation results, respectively and concat(·) means the concatenation.

3.5 Training Objectives

To train these modules in an end-to-end fashion, we need to combine two types of training losses. For the recommendation module, we optimize the cross-entropy loss as follows:

$$L_{rec} = -\frac{1}{M}\sum_{i=1}^{M} log\left(P_{rec}^i\right), \tag{21}$$

where M is number of ground truth recommendation items in conversation C. Pr_{rec}^i means the probability of the recommendation item i. For the generation module, we also employ the cross-entropy loss to optimize the module performance:

$$L_{gen} = -\frac{1}{N}\sum_{i=1}^{N} log\left(P_{gen}\left(t_i|t_1, \ldots, t_{i-1}\right)\right), \tag{22}$$

where N is the number of turns in a conversation C, t_i is the t-th utterance of this conversation. Then we integrate recommendation loss and generation loss with a hyperparameter γ:

$$L = L_{gen} + \gamma L_{rec} \tag{23}$$

4 EXPERIMENTAL

4.1 EXPERIMENTAL SETUP

In this section, we introduce the details of our experiment, including the dataset, baseline methods, evaluation metrics, and model implementation details.

4.1.1 Datasets

To evaluate the performance of our model, we utilize the REDIAL dataset [47] as the conversational recommendation dataset. This dataset is the most commonly used English dataset for conversational recommendation and generation tasks. REDIAL employs Amazon Mechanical Turk (AMT) to collect human conversations on movie recommendations, allowing researchers to systematically explore various model sub-components to address problems such as sentiment analysis and cold-start recommendation generation. It comprises 10,006 conversations focused on providing movie recommendations. After processing, we divide the dataset into three subsets with an 80:10:10 ratio for training, validation, and testing. As our method requires the dataset to include user IDs, currently only the Redial dataset meets our requirements.

4.1.2 Evaluation Metrics

Our goal is to recommend proper items to fulfill the needs of users and generate fluent sentences to answer their requests and chats. Therefore, we expect the results of our model to generate understandable statements for interaction with the user while recommending the correct results. To evaluate this task, we divide it into two parts for measuring, recommendation and generation parts.

For the recommendation task, we aim for the recommendation system's results to closely match those provided by human recommenders, which are more humanized and suitable for conversation. To evaluate the performance of our methods, we employ Recall@k (k=1,10,50), which indicates the percentage of top k items proposed by the system that include the ground truth item provided by human recommenders. Although a conversation typically only contains one or two items from recommendations, Recall@10 and Recall@50 demonstrate the effectiveness of the recommendation module. Recall@k is calculated as follows:

$$Recall@k = \frac{1}{N}\sum_{i \in N}\frac{S_{rec}^i \cap S_{human}^i}{S_{human}^i}, \tag{24}$$

where N represents the number of items that need to be recommended. S_{rec}^i denotes the top-k recommended items from the model, and S_{human}^i signifies the ground truth items recommended by human providers.

For the generation task, the goal of CRS is to seamlessly integrate appropriate items into the natural language response. Therefore, we also check whether the ground-truth item is included in the produced response in the end-to-end model by adopting Recall@k (k=1,10,50). Additionally, we use some metrics to evaluate the language performance of the final response. Following previous works, Distinct (Dist) n-gram (n=2,3,4) [48] can measure the diversity at the sentence level. These metrics are calculated as follows:

$$Dist(N) = \frac{\text{Count(different N-gram)}}{\text{Count(Total N-gram)}}, \tag{25}$$

where $Count(different\ N\text{-}gram)$ indicates the different continuous sequences of words in the sentence. $Count(Total\ N\text{-}gram)$ refers to the total continuous sequences in the sentence.

TABLE 1
Results of Recommendations. Our method shows statistically significant improvement compared to the baseline ($p < 0.05$).

Methods	Recall@1	Recall@10	Recall@50
TextCNN	0.013	0.068	0.189
ReDial	0.024	0.140	0.320
KBRD	0.032	0.150	0.336
KGSF	0.039	0.183	0.378
RevCore	0.061	0.236	0.454
CRFR	0.040	0.202	0.399
NTRD	0.031	0.186	0.397
C2-CRS	0.053	0.233	0.407
VRICR	0.057	0.251	0.416
PCHI(Ours)	**0.717**	**0.792**	**0.822**

TABLE 2
Experimental results on conversation task. Our method shows statistically significant improvement compared to the baseline($p < 0.05$). We abbreviate Distinct-2,3,4 as Dist-2,3,4.

Methods	Dist-2	Dist-3	Dist-4
Redial	0.225	0.236	0.228
KBRD	0.264	0.368	0.423
KGSF	0.289	0.434	0.519
NTRD	**0.578**	0.821	1.005
RID	0.518	0.624	0.598
BART	0.376	0.490	0.435
C2-CRS	0.163	0.291	0.417
VRICR	0.271	0.514	0.699
PCHI(ours)	0.574	**0.863**	**1.053**

4.1.3 Baselines

To evaluate the performance of item recommendation and response generation, we compare our model with the following methods.

(1) **Redial [47]:** This model consists of a hierarchical recurrent encoder following the HRED architecture [49] and a switching decoder. (2) **TextCNN [50]:** This model encodes utterances in the current session to learn user preferences by CNN-based mode. (3) **KBRD [4]:** This model has a knowledge-graph-enhanced recommendation architecture, which employs DBpedia to embed user preferences. The transformer-based architecture consists of the dialogue module, and KG information is presented as word bias for generation tasks. (4) **KGSF [5]:** This model utilizes two external knowledge graphs: a word-oriented KG and an item-oriented KG to further enrich the performance of the recommendation module. The generation model integrates the two KGs into decoders to obtain more detailed responses. (5) **NTRD [33]:** A Transformer-based model that generates responses using templates and item-slots, which can be filled in with the results from the recommendation module. (6) **RevCore [6]:** A model that employs a review-enhanced framework, using user's review information to improve the performance of both the recommender and dialogue generation components. (7) **CRFR [39]:** A conversational context-based reinforcement learning model with multi-hop reasoning on KGs. It flexibly learns multiple reasoning fragments which are likely contained in the complete paths of interests shifts. (8) **RID: [51]** A model combination the pre-trained language model (PLM) and an item-oriented knowledge graph. (9) **BART: [52]** A Large language model based on Transformer-based autoencoder. (10) **C2-CRS: [40]** A model with coarse-to-fine contrastive learning framework to improve data semantic fusion for CRS. (11) **VRICR: [53]** A model enhances incomplete KGs in CRSs by incorporating dialogue corpus and performs dynamic knowledge reasoning based on dialogue context.

TABLE 3
Comparison results on End-to-End recommendation.

Methods	Recall@1	Recall@10	Recall@50
KGSF	0.009	0.042	0.088
NTRD	0.018	0.125	0.316
RID	0.031	0.140	0.270
BART	0.017	0.071	0.138
GPT 3.5	0.034	0.172	-
PCHI(ours)	**0.454**	**0.473**	**0.493**

4.1.4 Experimental Settings

Our model is implemented in Pytorch and trained on a single NVIDIA 3080Ti 12G card. To facilitate faster recommendations, we assume that the last recommended item in each conversation is the item that the conversation truly intends to recommend, and we retain it during training. For the recommendation module, the item embedding size is set to 128, and we use the skip-gram algorithm to train Item2Vec with a negative sampling number of 5. Gradient clipping is applied to restrict gradients within the range [0, 0.1]. The batch size is set to 64. We employ the Adam optimizer with a learning rate of 5e-4. The weight γ is set to 5 when calculating the total loss. The source code developed for this study is available on GitHub

4.2 Performance Comparison of CRS

In this section, we begin by summarizing the comparison results for recommendation and response generation tasks separately. Then, we discuss whether our model can achieve accurate and personalized recommendations more quickly than other models.

4.2.1 Evaluation on Recommendation

Table 1 presents the results of the comparison experiments on the recommendation module. Our model PCHI outperforms the compared methods under all the metrics on the Redial dataset. Specifically, for Recall@1 metric, our method is 29.8 times larger than ReDial, 22.4 times larger than KBRD, 18.3 times larger than KGSF, 11.75 times larger than RevCore, 17.92 times larger than CRFR and 23.12 times larger than NTRD. For Recall@50 metric, our method is 156% higher than ReDial, 144% higher than KBRD, 117% higher than KGSF, 81% higher than RevCore, 106% higher than CRFR and 107% higher than NTRD. This indicates that our model can better capture user's preference information in the recommendation module. Moreover, this demonstrates that historical user interaction information can improve the accuracy of recommendations. On the other hand, We can observe that models such as KGSF and RevCore, introduce external information (e.g. KGs and user reviews) will perform better on recommendations compared to the original model (ReDial). It indicates that the external data can assist the model in extracting the user preference features during conversations, thus improving the recommendation accuracy. From the results, we can see that the recommendation performance of the model has been improved very significantly after combining the user's historical conversation information and knowledge graphs. We note that the biggest enhancement occurred in the top-1 metric, which means that our model is able to recommend results directly to the user, rather than letting the user choose from a large number of candidates.

4.2.2 Evaluation on Generation

Table 2 and Table 3 present the comparison results on Dist2/3/4 and End-to-End recall@1/10/50. Our model demonstrates effective performance on the Redial dataset and achieves the best results for most metrics, highlighting its ability to ensure diverse generation. In comparison to NTRD, our model performs similarly in the Dist-2 metric and outperforms NTRD in the end-to-end recommendation metrics, particularly in the recall@1 metric, where it provides superior top-1 recommendations. Significant gaps are observed between the accuracy of recommendations generated by end-to-end methods and recommendation modules in the end-to-end recommendation metrics. For instance, the recommendation based on the final produced response of the KGSF method only achieves 0.9% in Recall@1, much lower than the 3.9% achieved by the recommendation module. This discrepancy can be attributed to the susceptibility of KGSF's generation module to contextual interference and its limited effectiveness in embedding recommendation module results into dialogues. Even the relatively better-performing NTRD method experiences a significant loss of nearly 42% in the conversion between recommendation and generation. These results indicate the existing models' ineffectiveness in generating responses with accurate recommendations for users. Additionally, we also tested the performance of Chat-GPT in an end-to-end recommendation model. Based on the setup by Wang et al. [54], we used GPT 3.5 as our baseline model and tested the metrics recall@1 and recall@10. While Chat-GPT can generate very high-quality text and make recommendations based on user requests, it does not integrate well with user preferences, showing a significant performance gap on the redial dataset.

In contrast, our model excels in combining recommendation and generation, offering greater availability for end-to-end responses. By integrating user preference features and item features into the final generated results, our model achieves a recall@1 score of 45.4%. This finding suggests that the final responses do not require an excessive number of items to align with users' preferences, thereby enhancing recommendation accuracy.

4.3 Discussions

In addition to the performance comparison of baseline models, including Redial, KBRD, KGSF, NTRD, RID, RevCore, and CRFR, we also explore several variant models to demonstrate the importance of various components of our model and its ability to address specific problems. We discuss seven aspects of our experiments as follows: (1) The ablation study on recommendation module; (2) The impact of different methods in extracting historical information; (3) The impact of the improved structure on recommendation-to-generation loss; (4) The discussion about recommendation turns; (5) The discussion about parameters in end-to-end loss; (6) The impact of users' historical information on the model; (7) The discussion on user interest shift; (8) The case study. By analyzing these aspects, we can gain a deeper understanding of our model's strengths and weaknesses and further optimize its performance in recommendation and generation tasks.

4.3.1 The Ablation Study on Recommendation Module

We discussed the impact of different components in the recommendation module on model recommendation effectiveness and end-to-end recommendation effectiveness. We conducted ablation experiments using four model variants: (1) without external knowledge graph, (2) with only ConceptNet as our knowledge graph, (3) with only DBpedia as our knowledge graph, and (4) without attention mechanism. As the Table 4 shows, the variant of model which lacks an attention mechanism suffer significant performance losses in recommendation. Without utilizing the attention mechanism to condense

Fig. 3: Discussion on the impact of different methods in extracting historical information. The Rec. means the results of different model variants on the recommendation module, and Gen. means the recommendation results for different model variants in the final dialogue.

TABLE 4
The Ablation experiment in the recommendation module and end-to-end recommendation. R means Recall

	Recommendation			End-to-End		
Variations	R@1	R@10	R@50	R@1	R@10	R@50
No-KG	0.682	0.720	0.752	0.419	0.444	0.458
No-DB	0.730	0.766	0.793	0.452	0.467	0.487
No-Con	**0.750**	0.783	0.804	0.379	0.434	0.478
No-atten	0.083	0.197	0.325	0.016	0.087	0.243
PHCI	0.717	**0.792**	**0.822**	**0.454**	**0.473**	**0.493**

the vast information contained in the knowledge graph, and instead directly incorporating the obtained embeddings into our user embedding, the effectiveness of model recommendations will significantly decrease. On the other hand, the introduction of knowledge graphs has also enhanced the recommendation effectiveness of the model. In the variants of models without using knowledge graphs, both in terms of the efficiency of the recommendation module and in the end-to-end recommendation results, the complete PHCI model outperformed variants without knowledge graphs. Furthermore, even with just a single knowledge graph added, in terms of Recall@1 metric, models with DBpedia and ConceptNet outperformed the complete PCHI model. However, in PCHI, the integrating two knowledge graphs brought more insights to the model, resulting in better performance in recommendations for both top 10 and top 50 scenarios.

4.3.2 *The Impact of Different Methods in Extracting Historical Information*

To demonstrate the impact of different methods of extracting user history information features on the model, we perform experiments with variant models as follows:

1) W/o History Embedding: variant model without considering the history embedding.
2) RNN: variant model using Recurrent neural network in the model for history embedding.
3) Transformer: variant model using Transformer architecture in the model for history embedding.
4) Item2Vec: our PHCI model uses Item2Vec for history embedding.

The performance of different factors is illustrated in Fig.3. As observed, the model achieves better results when incorporating historical interaction information. Although the various model variants have different effects, they all demonstrate significant improvements compared to the model without history embedding. This highlights the effectiveness of the history embedding unit in learning users' preference information and the necessity of integrating users' long-term interests with their intentions expressed in the dialogue context. Additionally, we notice that the transformer-based model performs similarly to other models in the final results, despite not performing well in the recommendation module. This suggests that the transformer-based model has potential for end-to-end recommendation. To further understand the impact of historical information, we compare the proportion of different interaction counts in the dataset in Section 4.3.5.

4.3.3 *The Impact of the Improved Structure on Recommendation-to-Generation Loss*

Upon analyzing the results of the comparison experiments, it is evident that the baseline models experience significant performance loss when comparing the recommendation module and end-to-end recommendation results. This discrepancy can be attributed to various factors; for instance, the recommendation module may believe certain items should be recommended within a

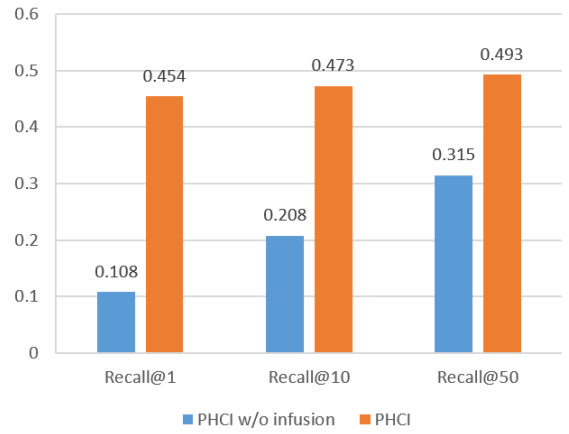

Fig. 4: Discussion on the impact of the improved structure on recommendation-to-generation loss, w/o infusion means use end-to-end results.

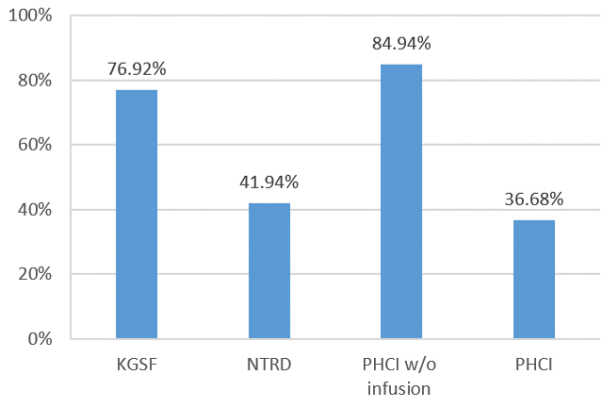

Fig. 5: R2G loss on different models, w/o infusion means use end-to-end results.

sentence and propose a set of items for the generation module. However, the generation module may decide to generate a response without a recommendation item to address the user's previous statement. To demonstrate the improvement in the generation module, we compare the performance of the improved generation module with a non-improved module. We refer to the model variant that directly employs end-to-end generation results as the w/o infusion model. In contrast, the concept involves outcomes derived from integrating templates of the generation module with items from the recommendation module. We have measured the recommendation efficacy and the conversion loss present in the results of both approaches.

The results are displayed in Fig.4. We can observe that the improved model demonstrates better performance when compared to the generation module without item embedding and preference features. It is hypothesized that the user preference feature and item feature can enhance the model's ability to predict the type of conversation to be generated, and aid the generation module in generating sentences with item slots for recommendations. We also notice that, compared to the unimproved model, the improved model exhibits smoother performance in Recall metrics. This suggests that item embedding and preference features can improve the accuracy of the model for end-to-end recommendation results and effectively reduce the size of the item candidate set. In addition, we compare these two variant models with baseline models and calculate the loss between their recommendation module and the final result.

As depicted in Fig.5, the improved model demonstrates the best performance among the baseline methods. We notice that the unimproved model has an 84.94% accuracy loss in recommendation-to-generation, which is higher than the baseline models. This indicates that without integrating item features and preference features into the generation module, the model cannot effectively convert good recommendation results into final results. We can also observe that the performance of NTRD and our model is better than KGSF, implying that models utilizing a slot-filling approach exhibit better performance in end-to-end recommendation results.

4.3.4 The Discussion about Recommendation Turns

In addition to high accuracy and diverse responses, we aim for users to receive satisfactory replies with fewer conversation turns. We designed a metric, the Average Turns for Recommendation (ATR), to calculate the average number of turns required to reach a final recommendation:

$$ATR = \frac{1}{n}\sum_{i \in n} Rank_i, \tag{26}$$

where n represents the number of successful recommendations, and $Rank_i$ denotes the turns that include the correct recommendation in a conversation. A smaller ATR indicates that users need fewer conversation turns to obtain their desired item. Furthermore, we employ the Mean Reciprocal Rank (MRR) to evaluate the model's speed:

$$MRR = \frac{1}{n}\sum_{i \in n} \frac{1}{Rank_i}, \tag{27}$$

The results are displayed in Table 5. We observe that the model incorporating user's historical interaction information outperforms the one that relies solely on context information. Moreover, we note that the baselines exhibit poor performance in MRR, which can be attributed to their weak Recall@1 when generating responses. This finding suggests that models adopting previous interaction data can effectively reduce the number of trial-and-error rounds in a conversation.

4.3.5 The Discussion about Parameters in End-to-End Loss

During end-to-end model training that combines the recommendation and generation modules, we introduce

TABLE 5
Eval on Recommendation Turns, ATR denotes average turns required for final results.

Methods	ATR	MRR
KGSF	6.20	0.001
NTRD	5.80	0.003
PCHI(Ours)	5.24	0.182

a hyperparameter γ. The purpose of γ is to determine the weights of the losses for both the recommendation and generation modules during end-to-end training. We conduct experiments to study the impacts of the hyperparameter γ, as shown in Table 6. We observe that the model's performance initially improves with increasing γ, but starts to decline when γ exceeds 5. Since the recommendation module has been pre-trained, a higher γ positively influences the final results. However, although the generation metrics improve as γ increases, an excessively large γ can have negative effects on the recommendation metrics.

TABLE 6
Discussion on the γ parameter

	Eval on Dialogue Gen.				Eval on end-to-end Rec.		
Parameters	PPL	Dist2	Dist3	Dist4	R@1	R@10	R@50
γ=0.5	6.467	0.497	0.718	0.847	0.406	0.448	0.479
γ=1	6.854	0.417	0.611	0.733	0.382	0.421	0.453
γ=3	6.431	0.505	0.729	0.881	0.381	0.415	0.453
γ=5	6.340	0.574	0.863	1.053	0.454	0.473	0.493
γ=10	6.190	0.481	0.735	0.987	0.408	0.449	0.489

4.3.6 *The Impact of User's Historical Information on Model*

In order to demonstrate the impact of historical information, we partition the dataset based on the number of user-item interactions. For example, if a user mentions an item only once in all their conversations, we label that as 1 user-item interaction. The results shown in Table 7 indicate that less than half of the users have only interacted with a particular movie once, 19% of users have interacted with a movie more than 5 times, and one user mentioned the same movie 64 times in all their conversations. From a data perspective, it is feasible to infer recommended items from movies users have historically interacted with, as multiple interactions suggest that users need recommendations for items similar to those from their history.

To further examine the influence of historical information on the model, we extracted users from the training set who had a history of interacting with specific movies and observed their performance in the test set. The results shown in Fig.6 reveal that as the number of user interactions increases, the model's performance in terms of final recommendations improves. However, when the number exceeds 10, the performance declines. This phenomenon can be explained by the fact that when a user mentions the same movie too many times, they likely have already seen the movie and know all related movies. In this case, recommendations similar to that movie can no longer meet the user's needs. This suggests that the model is more effective at capturing user preferences if a user mentions a movie more than five times in the dataset. Therefore, when utilizing a user's historical information, selecting an appropriate number of interactions as a prerequisite can better enhance the model's performance.

TABLE 7
Statistics on the number of user interactions with movies in Redial dataset

Numbers	1	2	3	4	5	6-10	10+
User count	346	133	41	45	24	101	37

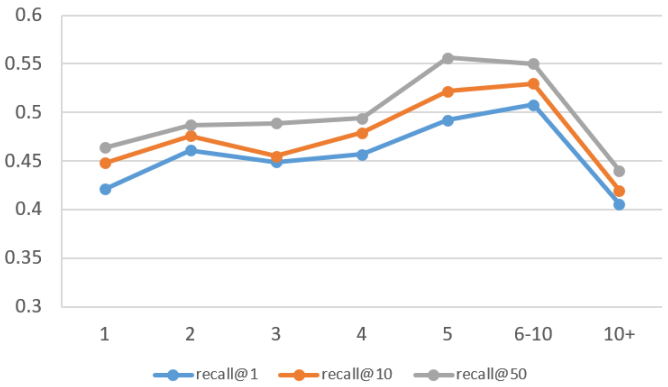

Fig. 6: Discuss on the impact of user's historical information on model

TABLE 8
Statistics on the occurrence of user interest shift in Redial dataset

	Number of Dialogue
User interest shift	196
No user interest shift	1146
Total	1342

4.3.7 *Discussion on User Interest Shift*

To further demonstrate our model's effectiveness in addressing the user interest shift problem, we compare its performance in the extraction test data with two baselines, KGSF and NTRD. The results for Recall@1, Recall@10, and Recall@50 are shown in Fig.7. We can observe that none of the three methods experienced significant performance loss in the presence of user interest shift. Except for KSGF, which already performed poorly in the end-to-end recommendation task, our method and

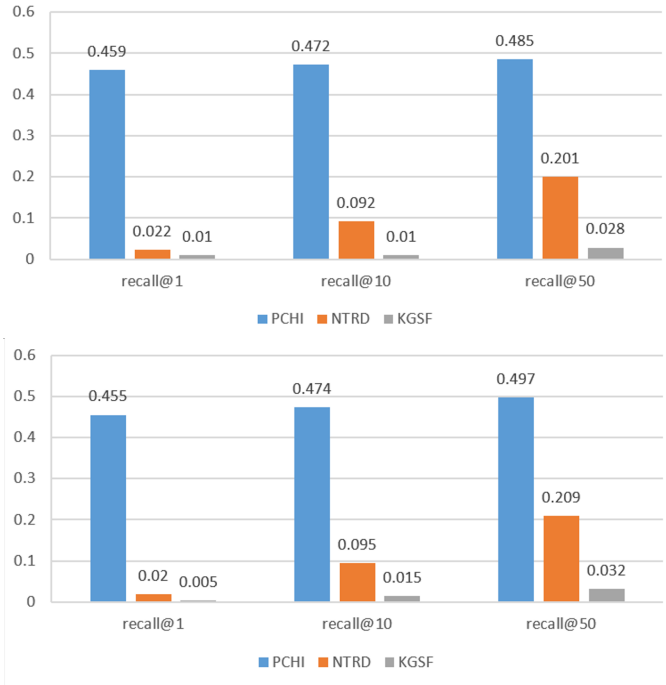

Fig. 7: Performance of different models on user interest transfer and no user interest transfer dataset, The upper section is a subdataset of user interest shifts, while the lower section is subdataset of no user interest shifts.

NTRD showed performance similar to that without user interest shift. This could be attributed to the fact that all methods are based on contextual information and exhibit strong locality, primarily relying on information from recent dialogues for recommendation. Therefore, our method is effective in handling user interest shift situation that frequently occurs in recommendation dialogues and provide new and precise recommendations to users.

4.3.8 Case Study

We display an example of responses predicted by our model and a human annotator for same dialogue in Fig.8. In the first round interaction, our model replies user's greeting and guiding user to say their needs. Then the user says he or she wants comedy movies like Blades of Glory. Our model captures the intention of "comedy movies" and "movies like Blades of Glory" and give the recommendation, shaun of the dead. After finding that the user is not satisfied with the movie, the model gives a new recommendation item until the user is satisfied. At the end of the conversation, our model is aware of the end of conversation and generates a goodwill response to the user.

5 Conclusion

In this paper, we propose a novel PCHI. Specifically, we introduce the user's historical interaction into traditional

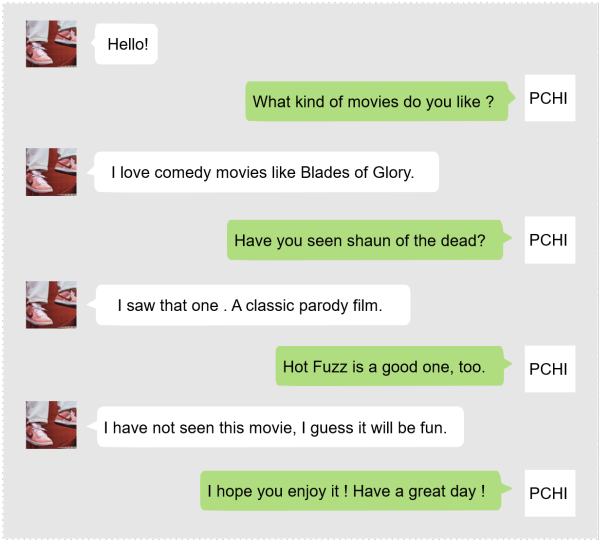

Fig. 8: An example dialogue with responses generated by PHCI and Human.

KG-enhanced CRS, by Item2Vec method and gate mechanism, which make the recommendation system more accurate and faster. To integrate the response generation and the item recommendation, we design a new fusion mechanism by incorporating the representation of recommendation items with the linguistic information given by the decoder. Extensive experiments on the benchmark dataset ReDial show our approach significantly outperforms the previous state-of-the-art methods in the recommendation task, and the diversity of final responses. In addition, we demonstrate the performance of the model on issues such as reducing conversation recommendation turns and shifting user interests. We also designed experiments to demonstrate the impact of different variants of the model on the recommendation and generation performance. In future work, we will look for better ways to integrate item information into the generated dialogues to generate dialogues that contain reasons for recommendations and characteristics of recommended items.

Acknowledgements

This work is in part funded by the NSFC, China under Grants 62372364; in part by China Postdoctoral Science Foundation (2020M683496); in part by Shaanxi Provincial Technical Innovation Guidance Plan, China under Grant 2024QCY-KXJ-199; in part by Key R&D Plan of Xianyang City, China under Grant L2023-ZDYF-QYCX-002; and in part by the National Postdoctoral Innovative Talents Support Program, China (BX20190273). (G. Zhao is the corresponding authors)

REFERENCES

[1] W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Conversational recommendation: Formulation, methods, and evaluation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2425–2428.

[2] D. Jannach, A. Manzoor, W. Cai, and L. Chen, "A survey on conversational recommender systems," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[3] W. Lei, X. He, Y. Miao, Q. Wu, R. Hong, M.-Y. Kan, and T.-S. Chua, "Estimation-action-reflection: Towards deep interaction between conversational and recommender systems," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 304–312.

[4] Q. Chen, J. Lin, Y. Zhang, M. Ding, Y. Cen, H. Yang, and J. Tang, "Towards Knowledge-Based Recommender Dialog System," in *EMNLP-IJCNLP*, 2019, pp. 1803–1813.

[5] K. Zhou, W. X. Zhao, S. Bian, Y. Zhou, J.-R. Wen, and J. Yu, "Improving conversational recommender systems via knowledge graph based semantic fusion," in *KDD '2020*, 2020, pp. 1006–1014.

[6] Y. Lu, J. Bao, Y. Song, Z. Ma, S. Cui, Y. Wu, and X. He, "RevCore: Review-Augmented Conversational Recommendation," in *ACL 2021*, 2021, pp. 1161–1173.

[7] C.-M. Wong, F. Feng, W. Zhang, C.-M. Vong, H. Chen, Y. Zhang, P. He, H. Chen, K. Zhao, and H. Chen, "Improving Conversational Recommender System by Pretraining Billion-scale Knowledge Graph," in *ICDE*, 2021, pp. 2607–2612.

[8] G. Penha and C. Hauff, "What Does BERT Know about Books, Movies and Music? Probing BERT for Conversational Recommendation," in *RecSys '2020*, 2020, pp. 388–397.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[10] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and W. B. Dolan, "DIALOGPT: Large-Scale Generative Pre-training for Conversational Response Generation," in *ACL 2020*, 2020, pp. 270–278.

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *ACL 2020*, 2020, pp. 7871–7880.

[12] Z. Wang, W. Wei, G. Cong, X.-L. Li, X.-L. Mao, and M. Qiu, "Global context enhanced graph neural networks for session-based recommendation," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 169–178.

[13] Y. Zhang, L. Wu, Q. Shen, Y. Pang, Z. Wei, F. Xu, B. Long, and J. Pei, "Multiple choice questions based multi-interest policy learning for conversational recommendation," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2153–2162.

[14] Y. Deng, Y. Li, F. Sun, B. Ding, and W. Lam, "Unified conversational recommendation policy learning via graph-based reinforcement learning," 2021.

[15] C. Gao, W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Advances and challenges in conversational recommender systems: A survey," *AI Open*, vol. 2, pp. 100–126, 2021.

[16] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft, "Asking clarifying questions in open-domain information-seeking conversations," in *SIGIR*, 2019, pp. 475–484.

[17] Y. Sun and Y. Zhang, "Conversational Recommender System," in *SIGIR '18*, 2018, pp. 235–244.

[18] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft, "Towards conversational search and recommendation: System ask, user respond," in *CIKM 2018*, 2018, pp. 177–186.

[19] W. Lei, X. He, Y. Miao, Q. Wu, R. Hong, M.-Y. Kan, and T.-S. Chua, "Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems," in *WSDM '20*, 2020, pp. 304–312.

[20] S. Li, W. Lei, Q. Wu, X. He, P. Jiang, and T.-S. Chua, "Seamlessly unifying attributes and items: Conversational recommendation for cold-start users," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 4, pp. 1–29, 2021, publisher: ACM New York, NY.

[21] K. Luo, S. Sanner, G. Wu, H. Li, and H. Yang, "Latent Linear Critiquing for Conversational Recommender Systems," in *WWW '20*, 2020, pp. 2535–2541.

[22] X. Ren, H. Yin, T. Chen, H. Wang, N. Q. V. Hung, Z. Huang, and X. Zhang, "CRSAL: Conversational Recommender Systems with Adversarial Learning," *ACM Trans. Inf. Syst.*, 2020.

[23] K. Xu, J. Yang, J. Xu, S. Gao, J. Guo, and J.-R. Wen, "Adapting User Preference to Online Feedback in Multi-Round Conversational Recommendation," in *WSDM '21*, 2021, pp. 364–372.

[24] J. Zou, Y. Chen, and E. Kanoulas, "Towards question-based recommender systems," in *SIGIR 2020*, 2020, pp. 881–890.

[25] W. Lei, G. Zhang, X. He, Y. Miao, X. Wang, L. Chen, and T.-S. Chua, "Interactive path reasoning on graph for conversational recommendation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '20. Association for Computing Machinery, 2020, p. 2073–2083.

[26] K. Zhou, W. X. Zhao, H. Wang, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, "Leveraging historical interaction data for improving conversational recommender system," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2349–2352.

[27] H. Li, S. Sanner, K. Luo, and G. Wu, "A ranking optimization approach to latent linear critiquing for conversational recommender systems," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 13–22.

[28] J. Lu, X. Ren, Y. Ren, A. Liu, and Z. Xu, "Improving contextual language models for response retrieval in multi-turn conversation," in *SIGIR*, 2020, pp. 1805–1808.

[29] L. Wang, H. Hu, L. Sha, C. Xu, K.-F. Wong, and D. Jiang, "Fine-tuning Large-Scale Pre-trained Language Models for Conversational Recommendation with Knowledge Graph," *arXiv preprint arXiv:2110.07477*, 2021.

[30] H. Wang, M. Cui, Z. Zhou, G. P. C. Fung, and K.-F. Wong, "Topicrefine: Joint topic prediction and dialogue response generation for multi-turn end-to-end dialogue system," *arXiv preprint arXiv:2109.05187*, 2021.

[31] B. Yang, C. Han, Y. Li, L. Zuo, and Z. Yu, "Improving Conversational Recommendation Systems' Quality with Context-Aware Item Meta Information," *arXiv preprint arXiv:2112.08140*, 2021.

[32] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[33] Z. Liang, H. Hu, C. Xu, J. Miao, Y. He, Y. Chen, X. Geng, F. Liang, and D. Jiang, "Learning neural templates for recommender dialogue system," *arXiv preprint arXiv:2109.12302*, 2021.

[34] J. Zhang, Y. Yang, C. Chen, L. He, and Z. Yu, "Kers: A knowledge-enhanced framework for recommendation dialog systems with multiple subgoals," in *EMNLP 2021*, 2021, pp. 1092–1101.

[35] Y. Zhu, J.-Y. Nie, K. Zhou, P. Du, H. Jiang, and Z. Dou, "Proactive retrieval-based chatbots based on relevant knowledge and goals," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2000–2004.

[36] W. Ma, R. Takanobu, and M. Huang, "CR-Walker: Tree-Structured Graph Reasoning and Dialog Acts for Conversational Recommendation," in *EMNLP '2021*, 2021, pp. 1839–1851.

[37] R. Sarkar, K. Goswami, M. Arcan, and J. P. McCrae, "Suggest me a movie for tonight: Leveraging Knowledge Graphs for Conversational Recommendation," in *COLING '2020*, 2020, pp. 4179–4189.

[38] T. Zhang, Y. Liu, P. Zhong, C. Zhang, H. Wang, and C. Miao, "KECRS: Towards Knowledge-Enriched Conversational Recommendation System," *arXiv preprint arXiv:2105.08261*, 2021.

[39] J. Zhou, B. Wang, R. He, and Y. Hou, "CRFR: Improving Conversational Recommender Systems via Flexible Fragments Reasoning on Knowledge Graphs," in *EMNLP*, 2021.

[40] Y. Zhou, K. Zhou, W. X. Zhao, C. Wang, P. Jiang, and H. Hu, "C2-crs: Coarse-to-fine contrastive learning for conversational recommender system," *arXiv preprint arXiv:2201.02732*, 2022.

[41] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *MLSP*, 2016, pp. 1–6.

[42] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[43] H. Liu and P. Singh, "ConceptNet—a practical commonsense reasoning tool-kit," *BT technology journal*, vol. 22, no. 4, pp. 211–226, 2004, publisher: Springer.

[44] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018, pp. 593–607.

[45] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

[46] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A Structured Self-attentive Sentence Embedding," *CoRR*, 2017.

[47] R. Li, S. Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal, "Towards deep conversational recommendations," in *NIPS*, 2018, pp. 9748–9758.

[48] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A Diversity-Promoting Objective Function for Neural Conversation Models," in *NAACL-HLT*, 2016, pp. 110–119.

[49] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *CIKM*, 2015, pp. 553–562.

[50] Y. Kim, "Convolutional neural networks for sentence classification," 2014.

[51] L. Wang, H. Hu, L. Sha, C. Xu, K.-F. Wong, and D. Jiang, "Recindial: A unified framework for conversational recommendation with pretrained language models," 2022.

[52] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[53] X. Zhang, X. Xin, D. Li, W. Liu, P. Ren, Z. Chen, J. Ma, and Z. Ren, "Variational reasoning over incomplete knowledge graphs for conversational recommendation," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 231–239.

[54] X. Wang, X. Tang, W. X. Zhao, J. Wang, and J.-R. Wen, "Rethinking the evaluation for conversational recommendation in the era of large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 10 052–10 065.