Long Short-Term Planning for Conversational Recommendation Systems

Xian Li¹, Hongguang Shi¹, Yunfei Wang¹, Yeqin Zhang¹, Xubin Li¹, and Cam-Tu Nguyen¹

State Key Laboratory for Novel Software Technology, Nanjing University, China {a81257,dream,woilfwang,zhangyeqin,lixubin}@smail.nju.edu.cn {ncamtu}@nju.edu.cn

Abstract. In Conversational Recommendation Systems (CRS), the central question is how the conversational agent can naturally ask for user preferences and provide suitable recommendations. Existing works mainly follow the hierarchical architecture, where a higher policy decides whether to invoke the conversation module (to ask questions) or the recommendation module (to make recommendations). This architecture prevents these two components from fully interacting with each other. In contrast, this paper proposes a novel architecture, the long short-term feedback architecture, to connect these two essential components in CRS. Specifically, the recommendation predicts the long-term recommendation target based on the conversational context and the user history. Driven by the targeted recommendation, the conversational model predicts the next topic or attribute to verify if the user preference matches the target. The balance feedback loop continues until the short-term planner output matches the long-term planner output, that is when the system should make the recommendation.

Keywords: Conversational Recommendation Systems, Planning

1 Introduction

Traditional recommendation systems rely on user behavior history, such as ratings, clicks, and purchases, to understand user preferences. However, these systems often encounter challenges due to the data sparseness issue. Specifically, users typically rate or buy only a small number of items, and new users may not have any records at all. As a result, achieving satisfactory recommendation performance becomes difficult. Furthermore, traditional models struggle to address two critical questions without clear user guidance and proactive feedback: (a) What are users interested in? and (b) What are the reasons behind each system recommendation?

The emergence of conversational recommender systems (CRSs) has fundamentally transformed traditional recommendation methods. CRSs facilitate the recommendation of products through multi-turn dialogues, enabling users and the system to dynamically interact using natural language. In these dialogues, the system not only elicits a user's preferences but also provides explanations for

2

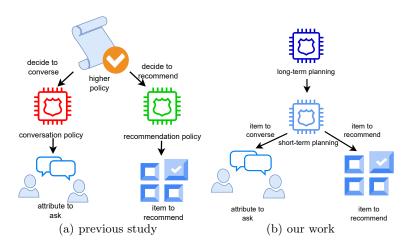


Fig. 1: The architecture of previous studies (a) and our work (b)

its recommended actions. Such capabilities are often absent in traditional recommendation approaches. Moreover, the conversational setting of CRSs presents a natural solution to the cold-start problem by allowing the system to proactively inquire about the preferences of new customers.

Most existing works for CRS use a hierarchical structure, where a higher policy determines whether to use the conversation module (to ask questions) or the recommendation module. This architecture prevents the conversation and the recommendation modules from fully interacting with each other. In contrast, this paper proposes a new approach where we plan the conversation and the recommendation by the same module, the short-term planner. Here, the short-term planner is influenced by a long-term planner that aims to model user long-term preference. Figure 1 demonstrates the main difference between our framework and the previous ones.

Our main constribution is three-fold: Firstly, it proposes a solution to combine user's past interactions and ongoing interactions (in conversations) into a long-term planner that takes into account the timestamps of these actions. Secondly, it presents a short-term planner that smoothly drives the conversations to the targeted item from the long-term planner. Lastly, it introduces a new dataset that captures practical aspects of both the recommendation module and the conversation module in the context of conversational recommender systems.

2 Related Work

Previous studies focus on the conversation (Light Conversation, Heavy Recommendation) or recommendation (Heavy Recommendation, Light Conversation).

2.1 Heavy Recommendation, Light Conversation

In this type of CRS, the aim is to understand user preferences efficiently and make relevant suggestions quickly. To achieve this, the system needs to focus on selecting the right attributes and asking the right questions. Sun et al. [16] and Lei et al. [7] proposed CRM and EAR, which train a policy of when and what attributes to ask. In such studies, the recommendation is made by an external recommendation model. Lei et al. [8] proposed SCPR that exploits a hierarchical policy to decide between asking and recommendation then invokes the corresponding components to decide what to ask and which to recommend Deng et al. [3] proposed UNICORN, a unified model to predict an item to recommend or an attribute to ask the question. The model allows rich interactions between the recommendation and the conversation model.

These methods rely on a simple conversation module that is limited to templated yes/no responses. Our approach differs from these studies as we focus on real conversations where users can actively change the dialog flow and agents should smoothly change the topic towards the targeted recommendation.

2.2 Light Recommendation, Heavy Conversation

This kind of CRS puts a greater emphasis on understanding conversations and generating reasonable responses. These methods [10,2,19,20,11,14] also adopt a hierarchical policy that decides between recommendation and conversation, but additional strategies are used to bridge the semantic gap between the word space of the conversation module and the item space of the recommendation module. Li et al. [10] proposed REDIAL that exploits a switching decoder to decide between recommendation and conversation. Chen et al. [2] proposed KBRD, that exploited a switching network like REDIAL [10], but improved the interactions between the recommendation and conversation modules with entity linking and a semantic alignment between a recommendation item to the word space for response generation. Zhou et al. [19] proposed KGSF that uses word-oriented and entity-oriented knowledge graphs (KGs) to enrich data representations in the CRS. They aligned these two semantic spaces for the recommendation and the conversation modules using Mutual Information Maximization. These methods (REDIAL, KGSF, KBRD) do not plan what to ask or discuss, but generate responses directly based on the conversational history. Zhou et al. [20] introduced TG-Redial where topic prediction is used as a planner for conversations. However, there is still a higher policy to decide to invoke the conversation or the recommendation module. Similarly, Zhang et al. [18] predicted a sequence of sub-goals (social chat, question answering, recommendation) to guide the dialog model. Here, the goal prediction plays the role of higher policy.

Unlike these methods, we do not have two distinct modules for deciding what to ask and what to recommend. Instead, we assume a knowledge graph (KG) that connects attributes, topics, and items, and aim to plan the next entity node on the graph for grounding the dialog. Specifically, a long-term policy module, which has access to user historical interactions, predicts a targeted item in the

KG. The short-term policy then exploits the targeted item and predicts either an attribute for conversation or an item for the recommendation. The objective of the short-term policy is to select a node in the KG so that the agent can smoothly drive the conversation to the long-term (targeted) item.

3 Preliminaries

Notations for CRS Formally, we are given an external knowledge graph \mathcal{G} that consists of an entity set V and a relation set \mathcal{E} (edges). The entity set \mathcal{V} contains all entities, including items and non-items (e.g., item attributes). Alternatively, the knowledge graph can be denoted as a set of triples (edges) $\{(e^h, r, e^t)\}$ where $e^h, e^t \in \mathcal{V}$ are the head and tail entities and r indicates the relationship between them.

A CRS aids users in purchase decisions via

Table 2: The description of the key symbols

| | Description |
|----------------------------|---|
| $\overline{\mathcal{P}^u}$ | The user profile. |
| \mathcal{C}^u | The current conversation. |
| \mathcal{S}^u | The entity sequence with timestamp derived |
| | from C^u and the user profile \mathcal{P}^u . |
| e^l | The targeted recommendation to be made in |
| | the upcomming turns (output of long-term |
| | plan). |
| e^s | The entity that will be grounded in upcomming |
| } | turns (output of short-term plan). |
| w | The latest user utterance in C^u . |
| \mathbf{z}^w | The representation of the current utterance. |
| \mathbf{z}^e | The representation of the current dialog entity |
| | sequence. |
| \mathcal{K} | Knowledge garph entities. |

conversation. During training, utterances from a user-agent conversation are labeled with entities from the knowledge graph \mathcal{K} . The agent's responses contain references to item entities for recommendations or non-item entities for clarification or chitchat. On the other hand, users refer to entities such as items or attributes to express their desired request.

In addition to the conversation history C, we also have access to the user profile for each user u. The user profile is a list of pairs (s_i, t_i) , where s_i is an item entity that the user u has interacted with in the past, and t_i is the time of the interaction. Here, the interactions can be purchases, browsing, or other historical activities besides conversations.

Task Definition Based on these notations, the CRS is defined as given a multitype context data (i.e., conversation, knowledge graph, user profile), we aim to (1) select entities from the knowledge graph to ground the next system response; (2) generate a response based on the selected entities. The selected entities may contain items (for the recommendation) or information about item attributes. The selected entities should be relevant to a dialog context (short-term user preference) and the user profile (long-term user preference).

4 Our Proposed Model

The LSTP's overall structure is depicted in Figure 2. The process begins with the extraction of entities from the conversation, of which the timestamps are set to zero. Then, the user profile is combined with this sequence to create a unified entity sequence. A targeted item entity is then selected by the long-term planner from the knowledge graph based on the sequence \mathcal{S}^u . Lastly, the short-term planner picks a grounding entity (item/non-item), which is then utilized to produce the next system response.

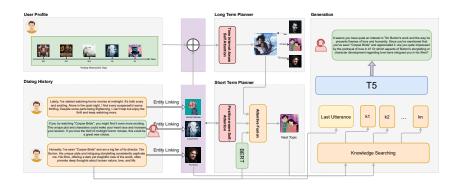


Fig. 2: The architecture of our LSTP framework

The long-term planner uses user profiles to more accurately target long-term user preferences, while the short-term planner considers recent utterances when planning the next entity in the conversation. The short-term planner is guided by the long-term planner to potentially lead to the long-term plan's target, while also ensuring the next entity is relevant to the dialogue history for a natural conversation. When a recommendation is necessary, the short-term planner should provide output consistent with the long-term planner.

4.1 Knowledge Representation and Grounding

Entities in the knowledge graph are represented by vectors in the same semantic space using Knowledge Graph Embeddings (KGE). This step is essential for both the long-term planner and the short-term one.

Knowledge Graph Embeddings. In this paper, we utilize TransE [1], which is available in the toolkit OpenKE[5], for knowledge graph embeddings. The main idea of TransE is that we represent entities and relations in the same semantic space so that if e^h should be connected to e^t via the relation r then $e^h + r \approx e^t$. Here, we use the same notation for entities (relations) and their vector representations. Formally, TransE learns a scoring function f as follows:

$$f(e^h, r, e^t) = -||e^h + r - e^t||_{1/2}$$

where $|\cdot|_{1/2}$ is either L_1 or L_2 norm, and $e^h, e^t \in \mathbb{R}^d$ with d = 1024 being the embedding dimention. The scoring function is larger if (e^h, r, e^t) is more likely to be a fact, i.e., e^h is connected to e^t via r in KG. Contrastive learning [5] is used to learn embeddings for all the entities and relations by enforcing the scores of true triples higher than those of negative (distorted) triples.

Entity Linking (or Knowledge Grounding). The objective of entity linking is to find entities previously mentioned in the dialog context. This is done by learning sentence representation so that it can be used to retrieve related entities from the knowledge graph. Specifically, we are given a training set of pairs (\mathbf{w}_i, e_i) in which \mathbf{w}_i indicates a conversational utterance and e_i is an entity mentioned in the utterance \mathbf{w}_i . User utterances are represented by BERT[4] whereas entities are represented as previously described. We exploit BERT large, and thus the output representation for the user utterance is of size 1204, which is the same with knowledge embeddings. Contrastive learning[5] is used to finetune the representation of the utterances \mathbf{w}_i so that the representation of \mathbf{w}_i is closer to the entity representation if e_i is mentioned in \mathbf{w}_i . Note that, here we only update the utterance representation while keeping entity embeddings unchanged.

4.2 Long-term Planning

The goal of the Long-term Planner (LTP) is to anticipate the upcoming recommendation that can be made based on a series of entities from the user profile and ongoing conversation context. To train LTP, we randomly select conversational contexts and their respective recommendations to create a dataset. It's important to note that not every conversation turn results in a recommendation, so the recommended item may come several turns after the latest turn in the dialog context. LTP is designed to consider a user's past interactions to make recommendations further in the future.

The training set for LTP consists of triples $(\mathcal{P}^u, C^u, e^l)$. Here, \mathcal{P}^u and C^u respectively represent the user profile and the context of the dialogue with the user u, and e^l indicates the targeted recommendation to be made in the upcoming turns. The entity sequence with timestamp derived from C^u and the user profile \mathcal{P}^u is denoted as $\mathcal{S}^u = \{(e_1^s, t_1), \dots, (e_l^s, t_l), (e_{l+1}^s, 0), \dots, (e_{l+m}^s, 0)\}$, where l and m respectively represent the count of entities in the user profile and dialog history. To ensure uniformity, we set the length of the sequence \mathcal{S}^u to be N and truncate the old entities in the sequence. The entity sequence can be then represented as $\mathcal{S}^u = \{(e_1^s, t_1), \dots, (e_N^s, t_N)\}$, where the timestamp t_i is zero if the corresponding entity is mentioned in the current dialog context instead of the user profile. Padding is applied to the sequences \mathcal{S}^u with length less than N.

We represent the sequence S^u by Multi-head Time-Aware Self-Attention (MH-TaSelfAttn) [9]. Unlike standard self-attention in Transformer[17], time-aware Self-attention (Ta-SelfAttn) takes into account the personalized interval between two user interactions (entities in S^u) to calculate the attention score between them. By personalization, the user-specific minimum and maximum interval values are considered for modeling temporal information between two user

interactions. Specifically, we initially represent each entity in S^u by knowledge embeddings, and then obtain the entity sequence representation as follows:

$$Z = MH-TaSelfAttn(S^u)$$
(1)

Here $Z = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ is the sequence of output representations and $\mathbf{z}_i \in R^d$. To predict the upcoming recommendation, we obtain the last vector from Z as the sequence representation \mathbf{z}^l . We then measure the relevance between \mathbf{z}^l and a candidate item using a dot product score. We then finetune the MH-TaSelfAttn layers to optimize the output representation so that the upcoming recommendation e^l is higher compared to other (negative) items. During inference, the item with the highest score \tilde{e}^l is used as the predicted recommendation target.

4.3 Short-term Planning

The purpose of the Short-term planner (STP) is to choose an entity that is related to the current dialog context and helps guide the conversation toward the LTP target. Intuitively, if the selected entity matches the LTP output, the next system response should provide a recommendation. The STP pays more attention to the current dialog when making decisions, unlike the LTP, which makes use of the user's historical actions. During STP training, the actual target (e^l) is used instead of the predicted (long-term) target \tilde{e}^l for the upcoming recommendation. In addition, STP accepts as input the lastest user utterance \mathbf{w} and the entity sequence \mathcal{S}^u_c , which is the part of \mathcal{S}^u containing entities in the dialog context C^u . The representation for multi-type input of STP is obtained by:

$$\mathbf{z}^w = BERT(\mathbf{w}) \tag{2}$$

$$\mathbf{z}^e = \text{Pooling}[\text{MH-SelfAttn}(\mathcal{S}_c^u)]$$
 (3)

$$\mathbf{z}^{s} = \text{Mean}[\text{SelfAttn}(\mathbf{z}^{w}, \mathbf{z}^{e}, e^{l})]$$
(4)

where the last equation shows how the short-term representation is obtained by fusing the current utterance representation \mathbf{z}^w , the current dialog entity sequence \mathbf{z}^e and the long-term target e^l . Here, Pooling indicates that we get the last item representation from MH-SelfAttn similarly to LTP, and SelfAttn indicates the standard self-attention operation in Transformer[17]. The STP is trained so that the next entity associated with the current context is higher compared to other entities. Like in LTP, only the additional layers in STP are finetuned, not entity embeddings. During inference, the item with the highest score \tilde{e}^s is used as the prediction for the next grounding entity.

4.4 Plan-based Response Generation

Given the next grounding entity \tilde{e}^s from the STP, and let \tilde{e}^s_{-1} be the grounding entity of the previous agent turn, knowledge search (Alg. 1) aims to select a set of surrounding K entities to(one or two hops away) improve the context for smooth response generation. This returned list is then flattened and combined with the

latest utterance w as input to the T5 model [13] for generating a response. During the training process, T5 is fine-tuned by optimizing the model to generate the correct response given the latest user utterance and correct grounding knowledge.

Algorithm 1 Knowledge Search

Require:

Grounding entities for the previous and next agent turns \tilde{e}_{-1}^s , \tilde{e}^s ; Knowledge Graph $\mathcal{K} = \{(e_k^h, r_k, e_k^t)\}_{k=1}^{N_{tri}}$ where e_k^h and e_k^t indicate the head and tail entities in the k-th triple;

Ensure:

```
Extended grounding knowledge list K
 1: Initialize K = \emptyset;
 2: for k = 1 to N_{tri} do
          if e_k^h = \tilde{e}_{-1}^s and e_k^t = \tilde{e}^s then
             K = K \cup (e_k^h, r_k, e_k^t)
 4:
          else if e_k^h = \tilde{e}_{-1}^s and \exists r \text{ so that } (e_k^t, r, \tilde{e}^s) \in \mathcal{K} then
 5:
             K = K \cup (e_k^h, r_k, e_k^t)
 6:
         else if e_k^h = \tilde{e}^s or e_k^t = \tilde{e}^s_{-1} then K = K \cup (e_k^h, r_k, e_k^t)
 7:
 8:
 9:
          end if
10: end for
11: return K[:20];
```

5 Data Collection

Our assumption is that a user's current preference should be influenced by their ongoing dialogue and long-term interests reflected by their historical actions. However, current datasets do not provide sufficient information for our evaluation. For instance, the ReDial dataset lacks user profiles. On the other hand, although the TG-Redial dataset offers user profiles, they are not accompanied by timestamps essential for LSTP modeling. Therefore, we created our dataset, TAP-Dial (Time-Aware Preference-Guided Dial). Data gathering for TAP-Dial is similar to TG-Redial but with some differences. Firstly, every user profile comes with a timestamp, which is not present in TG-Redial. Secondly, although the conversation grounding task in TAP-Dial resembles the next topic prediction in TG-Redial, we propose that there is a unified knowledge graph that links item attributes and topics. More details on our data collection are provided below.

Data Collection and Knowledge Graph Construction. We focused on movies as our domain of research and obtained raw data from the Douban website. Our selection process involved filtering users with inaccurate or irrelevant information to create a user set of 2,693. In addition, we gathered a total of 5,433 movies that were the most popular at the time as our item set. We also gathered supplementary data including information about directors, actors, tags, and reviews. A knowledge graph is then constructed with entities and relations as shown in Table 3.

| | Name | Number | Name | Number |
|----------|-----------------------------|--------|---------------------|--------|
| Entity | Movie | 5733 | Date | 8887 |
| | Star | 2920 | Number | 1223 |
| | Types of Movies | 31 | Key words | 2063 |
| | Location | 175 | Constellation | 12 |
| | Profession | 21 | Awards | 15816 |
| | The Constellation of | 2691 | The director of | 2766 |
| | The type of | 14566 | The release date of | 7862 |
| | The relative of | 470 | The country of | 842 |
| | The award records of | 35245 | The birth date of | 2607 |
| Relation | The popularity of | 5733 | The profession of | 8606 |
| | The key words of | 18369 | The birthplace of | 2852 |
| | The representative works of | 7668 | The score of | 5719 |
| | The screenwriter of | 2997 | Collaborate with | 1094 |
| | The main actors of | 14364 | Star | 14364 |

Table 3: Statistics of entities and relations in the knowledge graph

Dialog Flow Construction. To generate a list of recommendations for user conversations, we begin by selecting a set of targeted items. This is done by clustering the items in the user's history to determine a mixture of their preferences. We then choose the cluster centers as potential targets for recommendations, taking into account the most significant clusters and the timestamp.

Inspired by TG-Redial, we assume that the conversation should smoothly and naturally lead to the recommended items. Unlike TG-Redial, which relies on a separate topic set to ground non-recommendation turns, we use the knowledge graph's set of entities as potential grounding knowledge. In order to ensure smooth transitions between turns, we construct dialog flows consisting of lists of entities in the knowledge graph that gradually lead to the targeted items. Note that the first grounding entity can be randomly chosen.

Dialog Annotation. In the final stage, we recruit crowd-workers for writing the dialogs given dialog flows. We then received a total of 4416 dialogs for training, 552 dialogs for validation, and 552 for testing. Note that, all the dialogs are accompanied with grounding entities and targeted recommendations.

6 Experiments

6.1 Baselines and Metrics

In our experiments, we used several baselines, including REDIAL[10], KBRD[2], KGSF[19], TG-REDIAL[20]. All these baselines rely on sequence to sequence models as the bases for the conversation modules.

For evaluation, previous methods assume that there is an oracle policy that predefines recommendation turns, and evaluate the recommendation task and the conversation task separately. To ensure fairness, we compared our LSTP

method with other methods on the recommendation and conversation tasks separately using a similar procedure. We used MRR[15], NGCG[6], HIT for the recommendation task, and BLUE[12], Distinct, and F1 for the generation task.

6.2 Main Results

Recommendation The recommended task results are shown in Table4. It is observable that TG-REDIAL outperforms the other baselines. This is because it incorporates both contextual and historical sequence information. LSTP achieves the best performance as it includes not only sequence information but also temporal interval information. In the long-term plan-

Table 4: Results of recommendation task

| Model |] | NDCG | HIT | | |
|-----------|-------|-------|-------|-------|-------|
| Model | @1 | @10 | @50 | @10 | @50 |
| REDIAL | 0.002 | 0.010 | 0.048 | 0.005 | 0.013 |
| KBRD | 0.132 | 0.284 | 0.327 | 0.228 | 0.237 |
| KGSF | 0.103 | 0.222 | 0.263 | 0.177 | 0.186 |
| TG-REDIAL | 0.267 | 0.466 | 0479 | 0.399 | 0.404 |
| LSTP | 0.301 | 0.474 | 0.481 | 0.417 | 0.418 |

ning model, the accuracy can be improved by increasing the number of stacked attention modules, but it comes with time overhead. Therefore, a model with four stacked attention modules was chosen to balance between time and accuracy.

Dialog Generation The generated task results in Table5 indicate that LSTP performs the best. In comparison to the dialogue modules of other models, the advantage of LSTP lies in its ability to select relevant knowledge from the knowledge graph based on correct prediction results. Without the knowledge search module (LSTP w/o KS), the model's advantage would not be apparent, which also demonstrates the role of the Long-Short Term Planner (LSTP) module in predicting the next topic and making recommendations. Additionally, the distinctiveness of LSTP (w/o KS) is lower than the baseline, but the distinctiveness value of LSTP is significantly higher than the baseline. This confirms the significant impact of introducing external knowledge for diverse responses.

Table 5: The results of dialog generation task

| Model | BLEU@1 | BLEU@2 | BLEU@3 | BLEU@4 | Dist@1 | Dist@2 | Dist@3 | Dist@4 | F1 |
|---------------|--------|--------|--------|--------|--------|--------|--------|--------|------|
| REDIAL | 0.168 | 0.020 | 0.003 | 0.001 | 0.017 | 0.242 | 0.500 | 0.601 | 0.21 |
| KBRD | 0.269 | 0.070 | 0.027 | 0.011 | 0.014 | 0.134 | 0.310 | 0.464 | 0.28 |
| KGSF | 0.262 | 0.058 | 0.021 | 0.007 | 0.012 | 0.114 | 0.240 | 0.348 | 0.26 |
| TG-REDIAL | 0.183 | 0.040 | 0.013 | 0.005 | 0.013 | 0.153 | 0.352 | 0.532 | 0.22 |
| LSTP (w/o KS) | 0.297 | 0.106 | 0.054 | 0.029 | 0.019 | 0.182 | 0.332 | 0.450 | 0.32 |
| LSTP | 0.333 | 0.136 | 0.081 | 0.054 | 0.022 | 0.263 | 0.519 | 0.607 | 0.38 |

6.3 Additional Analysis

Conversational Grounding Task Following TG-Redial [20], we compare LSTP to several baselines including MGCG/11, Connv-Bert, Topic-Bert [20] on predicting

Table 6: Grounding entity prediction at non-recommendation turns

| Model | HIT@1 | HIT@3 | HIT@5 |
|------------|-------|-------|-------|
| Conv-Bert | 0.169 | 0.245 | 0.285 |
| Topic-Bert | 0.251 | 0.348 | 0.394 |
| MGCG | 0.174 | 0.281 | 0.335 |
| TG-REDIAL | 0.219 | 0.327 | 0.382 |
| LSTP | 0.312 | 0.447 | 0.482 |

Table 7: HIT@1 of LSTP with variants of the Short-term planner

| | Over. | | |
|------------------|-------|-------|-------|
| LSTP | 0.308 | 0.301 | 0.312 |
| w history | 0.279 | 0.33 | 0.254 |
| w/o long-term | 0.304 | 0.286 | 0.312 |
| w/o lastest utt. | 0.308 | 0.305 | 0.309 |

entities to ground the conversation at non-recommendation turns. The results of the entity prediction for non-recommendation turns are shown in Table6, demonstrating that the LSTP model achieves the best performance. This is partially because LSTP incorporates not only sequence information but also temporal interval information and sentence information.

Ablation Study The results of our ablation study are presented in Table 7. Here, Over., Rec., Conv. respectively refer to the grounding prediction at all the turns, recommendation turns, or conversation turns. We implemented the Long Short-Term Planning (LSTP) with different variants of the short-term planner, where we include history to the short-term planner (w/ history), exclude the long-term planner (w/o long-term) or the latest user utterance (w/o latest utt). When integrating the user's historical interaction into the short-term planner, we observed a substantial enhancement in the recommendation outcomes but a significant deterioration in the conversation results. On the other hand, without the guidance of long-term planning (w/o long-term), the performance of recommendations suffered, partially demonstrating the importance role of the long-term planner. In contrast, the consideration of the latest user utterance did not seem to have a significant impact on the results, partially showing that entity linking might provide sufficient information for planning.

7 Conclusion

In this paper, we investigated the issue of the insufficient interaction between the dialogue and recommendation modules in previous CRS studies. We introduced LSTP model, which consists of a long-term model and a short-term module. The long-term model predicts a targeted recommendation based on long-term human interactions (historical interactions). The short-term model is able to predict the subsequent topic or attribute, thereby ascertaining if the user's preference aligns with the designated target. This harmonious feedback loop is continuously cycled until the output from the short-term planner matches the long-term planner's output. The equilibrium state indicates the system's optimal readiness for recommendation. We crafted a novel conversation dataset reflecting this dynamic. Experimental results on this dataset verify the effectiveness of our method.

Acknowledgements

We thank the data annotators for their meticulous work on dialogue annotation, which was pivotal for this research.

References

- Antoine Bordes, Nicolas Usunier, et al. Translating embeddings for modeling multi-relational data. NIPS, 2013.
- Qibin Chen, Junyang Lin, et al. Towards knowledge-based recommender dialog system. EMNLP, 2019.
- 3. Yang Deng, Yaliang Li, Fei Sun, et al. Unified conversational recommendation policy learning via graph-based reinforcement learning. In SIGIR, 2021.
- 4. Jacob Devlin, Ming-Wei Chang, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. NAACL, 2018.
- 5. Xu Han, Shulin Cao, et al. Openke: An open toolkit for knowledge embedding. In EMNLP, 2018.
- Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. TOIS, 2002.
- Wenqiang Lei, Xiangnan He, et al. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In WSDM, 2020.
- 8. Wenqiang Lei, Gangyi Zhang, et al. Interactive path reasoning on graph for conversational recommendation. In KDD, 2020.
- 9. Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In WSDM, 2020.
- 10. Raymond Li, Samira Ebrahimi Kahou, et al. Towards deep conversational recommendations. In NIPS, 2018.
- 11. Zeming Liu, Haifeng Wang, et al. Towards conversational recommendation over multi-type dialogs. In ACL, 2020.
- 12. Kishore Papineni, Salim Roukos, et al. Bleu: a method for automatic evaluation of machine translation. In ACL, 2002.
- 13. Colin Raffel, Noam Shazeer, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 2020.
- Xuhui Ren, Hongzhi Yin, et al. Learning to ask appropriate questions in conversational recommendation. In SIGIR, 2021.
- Gerard Salton and Michael McGill. Introduction to modern information retrieval. 1984.
- 16. Yueming Sun and Yi Zhang. Conversational recommender system. In SIGIR, 2018.
- 17. Ashish Vaswani, Noam Shazeer, et al. Attention is all you need. NIPS, 2017.
- Jun Zhang, Yan Yang, et al. KERS: A knowledge-enhanced framework for recommendation dialog systems with multiple subgoals. In *Findings of EMNLP*, 2021.
- Kun Zhou, Wayne Xin Zhao, et al. Improving conversational recommender systems via knowledge graph based semantic fusion. In KDD, 2020.
- Kun Zhou, Yuanhang Zhou, et al. Towards topic-guided conversational recommender system. In COLING, 2020.