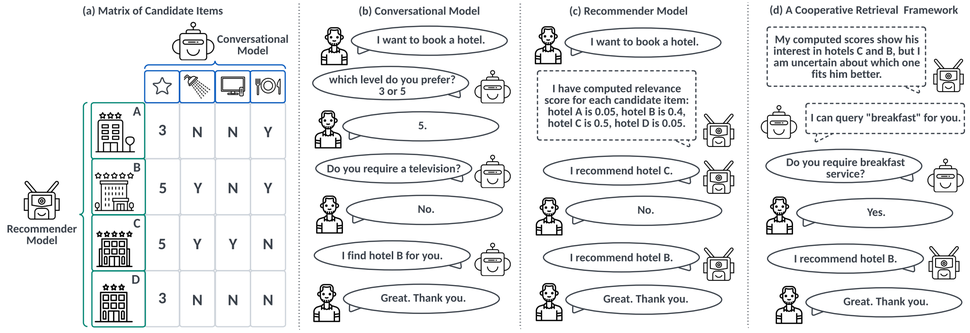
Lending Interaction Wings to Recommender Systems with Conversational Agents

Jiarui Jin[1], Xianyu Chen[1], Fanghua Ye[2], Mengyue Yang[2], Yue Feng[2], Weinan Zhang[1], Yong Yu[1], Jun Wang[2]

[1]Shanghai Jiao Tong University, [2]University College London

Background



Our main idea is to bridge the conversational model's ability of online querying user preference on each attribute and the recommendation model's ability of offline estimating user interest for each item, allowing the whole system to query either an attribute or an item to users.

Challenges

The main challenges here could be summarized into two aspects. One is about the system design (i.e., how to bridge querying attributes and querying items), and the other one is about the methodology design (i.e., how to develop a unified metric for evaluations of attributes and items).

- (a) Prior literature directly combine the conversational models and the recommender models with a reinforcement learning based framework. However, reinforcement learning based methods mainly suffer from data insufficiency problem and their training procedure usually requires a large amount of data.
- (b) Previous solution for determining whether items or attributes is to establish a new neural network as a controller to control when to use the recommender component and when to apply the conversational component. However, they lack a unified metric to mathematically measure whether to query an item or an attribute in each case.

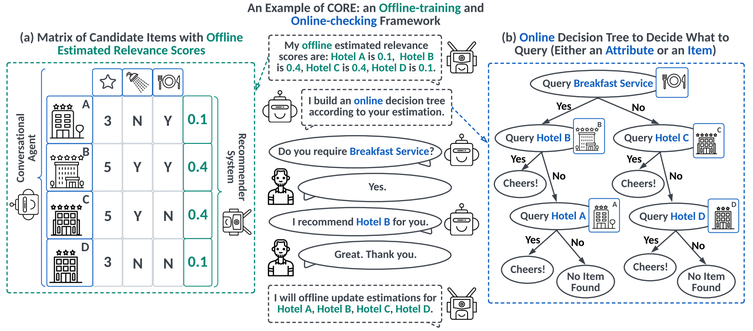
Pros of Our Idea

For the system design, our idea only needs the API of the conversational model and the API of the recommender model For the methodology design, our idea is to define a new evaluation metric called certainty gain over each possible item and attribute.

- (a) Since our method only requires API of either model, the recommender model could be offline tuned with historical logs of users while the conversational model could be any of existing large language model (because many best-of-class large language models are only accessible through API).
- (b) We define uncertainty as the summation of the remaining estimated relevance scores, and then we can derive certainty gain as the change of querying each attribute or each item. Based on this, we construct an online decision tree algorithm, which can be built in a non-parametric style. Hence, the choice of each item or each query has its own theoretical support.

As a consequence, our model is using the recommender systems as an offline estimations (estimating the relevance scores according to the user's historical data as the prior) while using the conversational agent as an online checker (checking whether our offline estimations fit the user online preference). Here, large language models play two roles: (i) answer extractor: encoding the context of conversations and (ii) question generator: generating new conversation turns.

Our Framework (CORE)



CORE, an offline-training and online-checking framework, where a recommender system operates as an offline relevance score estimator (colored in green), while a conversational agent acts as an online relevance score checker (colored in blue). Concretely, given a matrix of candidate items, as shown in (a), the recommender system could offline assign an estimated relevance score to each item, and then the conversational agent would online check these scores by querying either items or attributes, depicted in (b).

Our key idea is to define the uncertainty as the summation of the relevance scores are still unchecked. Then, it is easily to derive the certainty gain of querying each attribute or item as the relevance scores we can removed by querying each attribute or item.

Definition 1 (Uncertainty and Certainty Gain). *For the k-th turn, we define uncertainty, denoted as $\mathtt{U}_k$, to measure how many estimated relevance scores are still unchecked, i.e.,*

$$\mathtt{U}_k := \mathtt{SUM}(\{\Psi_{\mathtt{RE}}(v_m) | v_m \in \mathcal{V}_k\}), \tag{1}$$

where $\Psi_{\mathtt{RE}}(v_m)$[1] outputs the estimated relevance score for item v_m, $\mathcal{V}_k$ is the set of all the unchecked items after k interactions, and $\mathcal{V}_k$ is initialized as $\mathcal{V}_0 = \mathcal{V}$. Then, the certainty gain of k-th interaction is defined as $\Delta\mathtt{U}_k := \mathtt{U}_{k-1} - \mathtt{U}_k$, i.e., how many relevance scores are checked at the k-th turn. Since our goal is to find a target item, if $\Psi_{\mathtt{CO}}(\cdot)$ successfully finds one at the k-th turn, then we set all the items checked, namely $\mathtt{U}_k = 0$.

In this regard, our conversational agent is to minimize $\mathtt{U}_k$ at each k-th turn via removing those checked items from $\mathcal{V}_k$. Considering that online updating $\Psi_{\mathtt{RE}}(\cdot)$ is infeasible in practice due to the high latency and computation costs, the objective of $\Psi_{\mathtt{CO}}(\cdot)$ can be expressed as:

$$\min_{\Psi^*_{\mathtt{RE}}} K, \text{ s.t., } \mathtt{U}_K = 0, \tag{2}$$

where K is the number of turns, and $\Psi^*_{\mathtt{RE}}$ means that the recommender system is frozen. To this end, the design of our conversational agent could be organized as an uncertainty minimization problem.

Therefore, the resulting online decision tree algorithm is to greedily choice an attribute or an item that can maximize the certainty gain: $a_{\mathtt{query}} = \underset{a \in \mathcal{V}_{k-1} \cup \mathcal{X}_{k-1}}{\arg\max} \Psi_{\mathtt{CG}}(\mathtt{query}(a))$, Concretely, the formulation of querying an attribute can be written as: $\Psi_{\mathtt{CG}}(\mathtt{query}(a)) = \sum_{w_a \in \mathcal{W}_a} \left(\Psi_{\mathtt{CG}}(w_a = w_a^*) \cdot \Pr(w_a = w_a^*)\right)$, where $w_a = w_a^*$ means that when querying a, the user's answer (represented by w_a^* is w_a).

The formulation of querying an item is $\Psi_{\mathtt{CG}}(\mathtt{query}(a)) = \Psi_{\mathtt{CG}}(a \in \mathcal{V}^*) \cdot \Pr(a \in \mathcal{V}^*) + \Psi_{\mathtt{CG}}(a \notin \mathcal{V}^*) \cdot \Pr(a \notin \mathcal{V}^*)$ where $a \in \mathcal{V}^*$ means that queried a is a target item. $= \left(\sum_{v_m \in \mathcal{V}_{k-1}} \Psi_{\mathtt{RE}}(v_m)\right) \cdot \Pr(a \in \mathcal{V}^*) + \Psi_{\mathtt{RE}}(a) \cdot \Pr(a \notin \mathcal{V}^*)$,

Algorithm

Our paper lists two separate configurations in terms of querying attributes: one is directly querying attribute IDs (e.g., querying a user what level of hotel she prefers), while the other one is query attribute values (e.g., querying a user whether she prefers hotels with level 3).

Algorithm 1 CORE for Querying Items and Attributes

Input: A recommender system $\Psi_{\mathtt{RE}}(\cdot)$, an item set $\mathcal{V}$, an attribute set $\mathcal{X}$, an offline dataset $\mathcal{D}$.
Output: Updated recommender system $\Psi_{\mathtt{RE}}(\cdot)$, up-to-date dataset $\mathcal{D}$.

```
Train Ψ_RE(·) on D.                                          ▷ Offline-Training
for each session (i.e., the given user) do
    Initialize k = 1 and V_0 = V, X_0 = X.
    repeat
        Compute a_query following Eq. (3) for querying items and attributes or following Eq. (10)
        for querying items and attribute values.
        Query a_query to the user and receive the answer.    ▷ Online-Checking
        Generate V_k and X_k from V_{k-1} and X_{k-1}.       ▷ Online-Checking
        Go to next turn: k ← k + 1.
    until Querying a target item or k > K_MAX where K_MAX is the maximal number of turns.
    Collect session data and add to D.
end for
Update Ψ_RE(·) using data in D.                              ▷ Offline-Training
```

Empirical Evidence

Table 2: Results comparison of querying attribute values on tabular datasets. See Table A2 for the full version.

$\Psi_{\mathtt{RE}}(\cdot)$	$\Psi_{\mathtt{CO}}(\cdot)$	Amazon				LastFM				Yelp			
		T@3	S@3	T@5	S@5	T@3	S@3	T@5	S@5	T@3	S@3	T@5	S@5
COLD START	AG	6.47	0.12	7.83	0.23	6.77	0.05	8.32	0.14	6.65	0.08	8.29	0.13
	ME	6.50	0.12	8.34	0.16	6.84	0.04	8.56	0.11	6.40	0.15	8.18	0.20
	CORE	**6.02**	**0.25**	**6.18**	**0.65**	**5.84**	**0.29**	**5.72**	**0.74**	**5.25**	**0.19**	**6.23**	**0.65**
	$\text{CORE}^+_{\mathtt{O}}$	6.00	0.26	6.01	0.67	5.79	0.30	5.70	0.75	5.02	0.21	6.12	0.68
FM	AG	**2.76**	0.74	**2.97**	0.83	4.14	0.52	4.67	0.64	3.29	0.70	3.39	0.81
	CRM	4.58	0.28	6.42	0.38	4.23	0.34	5.87	0.63	4.12	0.25	6.01	0.69
	EAR	4.13	0.32	6.32	0.42	4.02	0.38	5.45	0.67	4.10	0.28	5.95	0.72
	CORE	3.26	**0.83**	3.19	**0.99**	**3.79**	**0.72**	**3.50**	**0.99**	**3.14**	**0.84**	**3.20**	**0.99**
	$\text{CORE}^+_{\mathtt{O}}$	3.16	0.85	3.22	1.00	3.75	0.74	3.53	1.00	3.10	0.85	3.23	1.00
DEEP FM	AG	**3.07**	0.71	3.27	0.82	3.50	0.68	3.84	0.79	3.09	0.74	3.11	0.88
	CRM	4.51	0.29	6.32	0.40	4.18	0.38	5.88	0.63	4.11	0.23	6.02	0.71
	EAR	4.47	0.30	6.35	0.43	4.01	0.37	5.43	0.69	4.01	0.32	5.74	0.75
	CORE	3.23	**0.85**	**3.22**	**0.99**	**3.47**	**0.81**	**3.34**	**1.00**	**2.98**	**0.93**	**3.11**	**1.00**
PNN	AG	3.02	0.74	3.10	0.87	3.44	0.67	3.53	0.84	2.83	0.77	2.82	0.91
	CORE	**3.01**	**0.88**	**3.04**	**0.99**	**3.10**	**0.87**	**3.20**	**0.99**	**2.75**	**0.88**	**2.76**	**1.00**

Table 1: Results comparison in the context of querying attributes. See Table A1 for the full version.

$\Psi_{\mathtt{RE}}(\cdot)$	$\Psi_{\mathtt{CO}}(\cdot)$	Amazon			
		T@3	S@3	T@5	S@5
COLD START	ME	3.04	0.98	5.00	1.00
	CORE	**2.88**	**1.00**	**2.87**	**1.00**
	$\text{CORE}^+_{\mathtt{O}}$	2.84	1.00	2.86	1.00
FM	AG	2.76	0.74	2.97	0.83
	CRM	3.07	0.98	3.37	1.00
	EAR	2.98	0.99	3.13	1.00
	CORE	**2.17**	**1.00**	**2.16**	**1.00**
	$\text{CORE}^+_{\mathtt{O}}$	2.14	1.00	2.14	1.00

Links

Paper Link

Project Link

If you have any question, please feel free to contact Jiarui Jin (jinjiarui97@gmail.com)