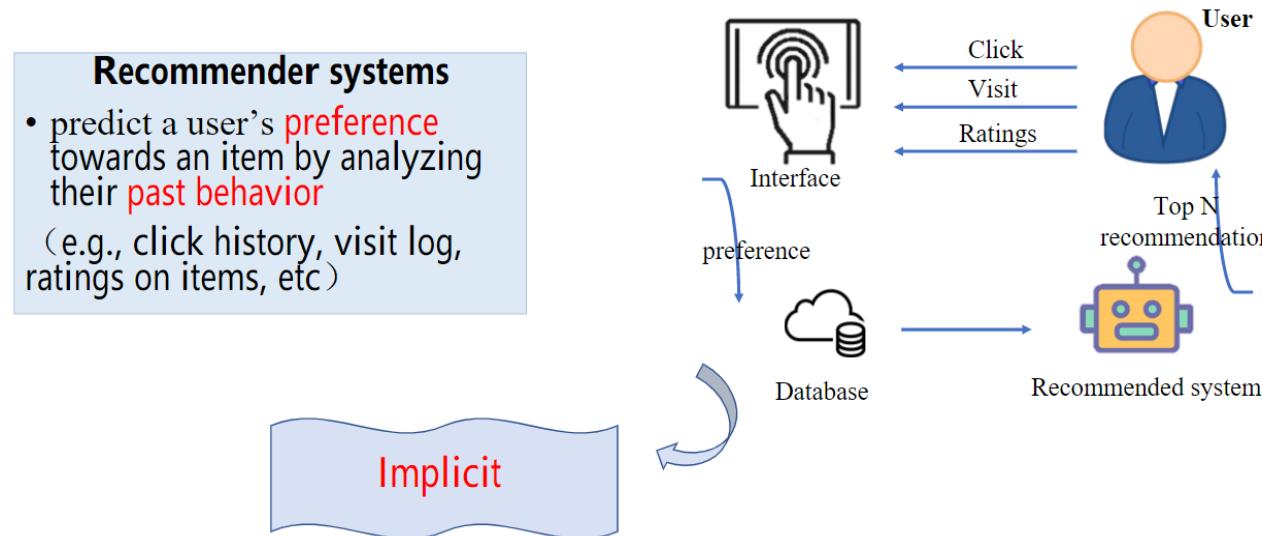


Conversational Recommendation

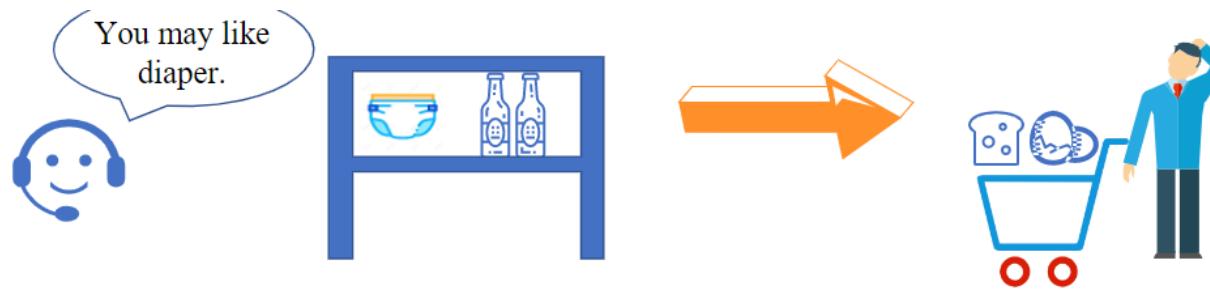
Zijing Yang

Typical Recommender Systems



Limitation:

- Users' preferences often drift over times
- It is hard to find accurate reasons to recommendation



Conversation Brings Revolution

Can you find me a **mobile phone** on Amazon?
Sure, what **operating system** do you prefer? 🤖
I want an **Android** one.
OK, and any preference on **screen size**? 🤖
Better larger than **5 inches**.
Do you have requirements on **storage capacity**? 🤖
I want it to be at least **64 Gigabytes**.
And any preference on **phone color**? 🤖
Not particularly.
Sure, then what about the following choices? 🤖

I don't like them very much...
OK, do you have any preference on the **brand**? 🤖
Better be **Samsung** or **Huawei**.
Any requirement on **price**? 🤖
Should be **within 700 dollars**.
OK, then what about these ones? 🤖

Great, I want the first one, can you order it for me?
Sure, I have placed the order for you, enjoy! 🎉

Hi! I'm looking for a **dance** music artist.
Do you like **rock** music? 🤖
Yes! I like it!
Do you like **pop** music? 🤖
Yes! I like it!
You may like music artist **Michael Jackson**!
Yes! Thank you!

The example of a conversational recommender system

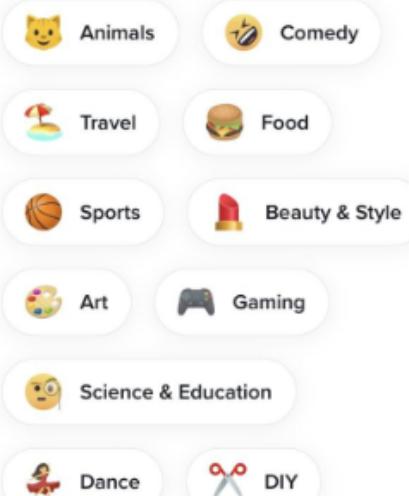
Conversational Recommender Systems

➤ Interactive recommendation

➤ Using natural languages

Choose your interests

Get personalized video recommendations



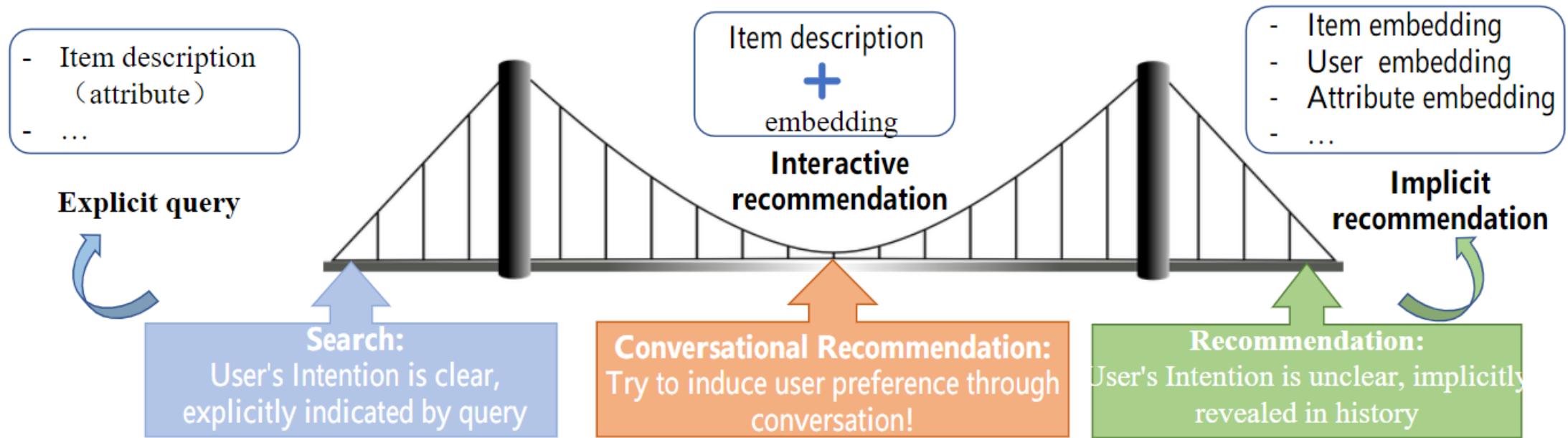
Next

The example of tag-based interaction on tiktok

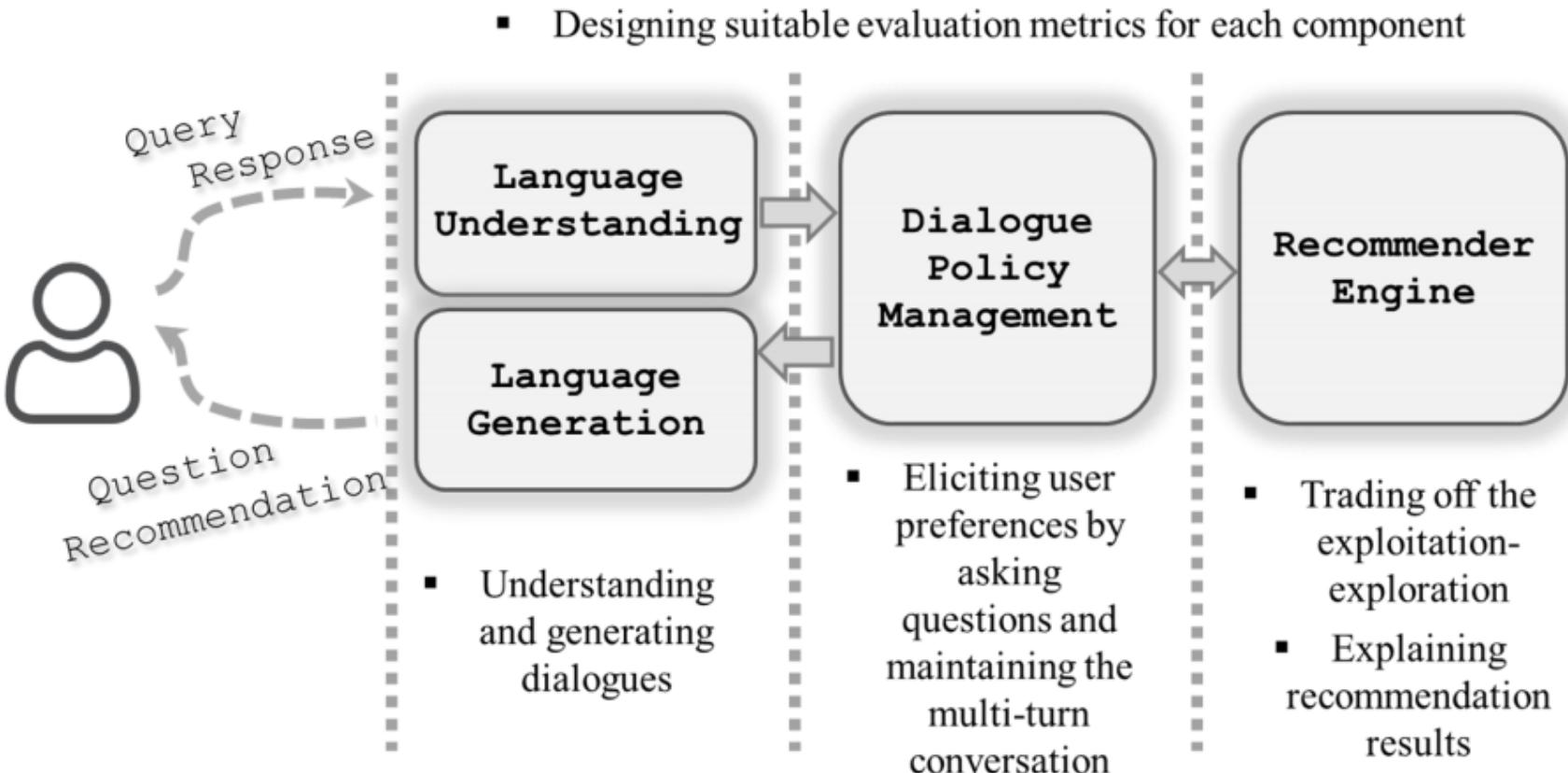
Conversational Recommendation

Bridges Search and Recommendation

Traditional paradigms for information-seeking:
Search (pull) or Recommendation (push)



Conversational Recommender Systems



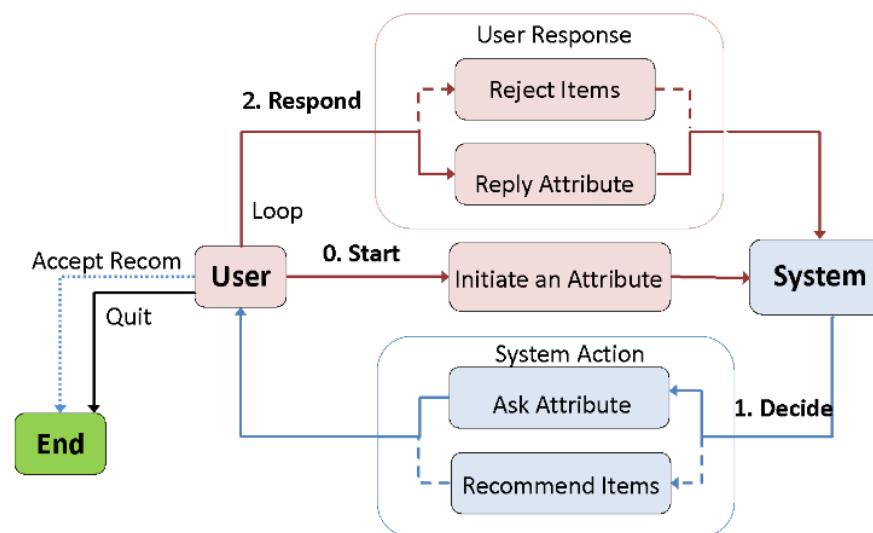
Multi-turn Conversational Recommendation

A System needs to choose to **ask questions** and **make recommendations** in a multi-turn conversation

□ **Purpose:** making successful recommendations with less turns of interactions

□ **Challenges to address:**

1. Which items or attributes to recommend?
2. When to ask questions and when to make recommendations?
3. How to adapt user feedback



Lei et al. "Estimation–Action–Reflection: Towards Deep Interaction Between Conversational and Recommender Systems" (WSDM'20)

Evaluation Metrics:

Evaluation Matrices:

- SR @ k (Success rate at k-th turn)
- AT (Average Turns)

$$SR = \frac{\#successful\ dialogues}{\#dialogues} \cdot 100\%$$

$$AT = \overline{dialogue\ length}$$

Conversational Recommender System

- Shortcoming: only utilize user **inputs in the current session**, ignoring **long term preferences**
 - NLU: understanding, dialogue history, updating intention —> extracting item meta data
 - DM: decides which action to take given the current state
 - Generator: generate response
- NLU: belief tracker—>update user intention
 - DM: policy network (queries, preferences)—>action
 - Generator: generate response

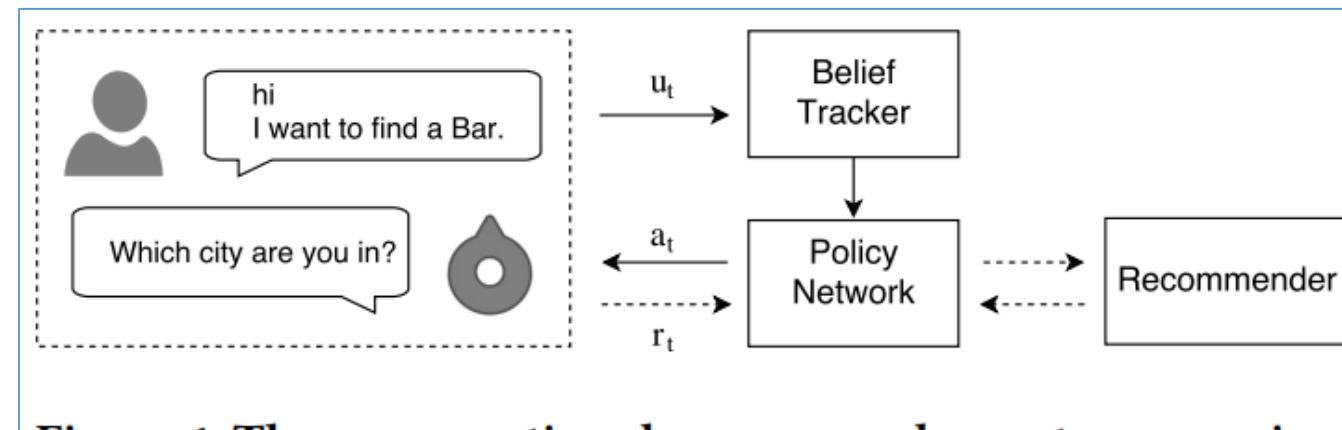


Figure 1: The conversational recommender system overview

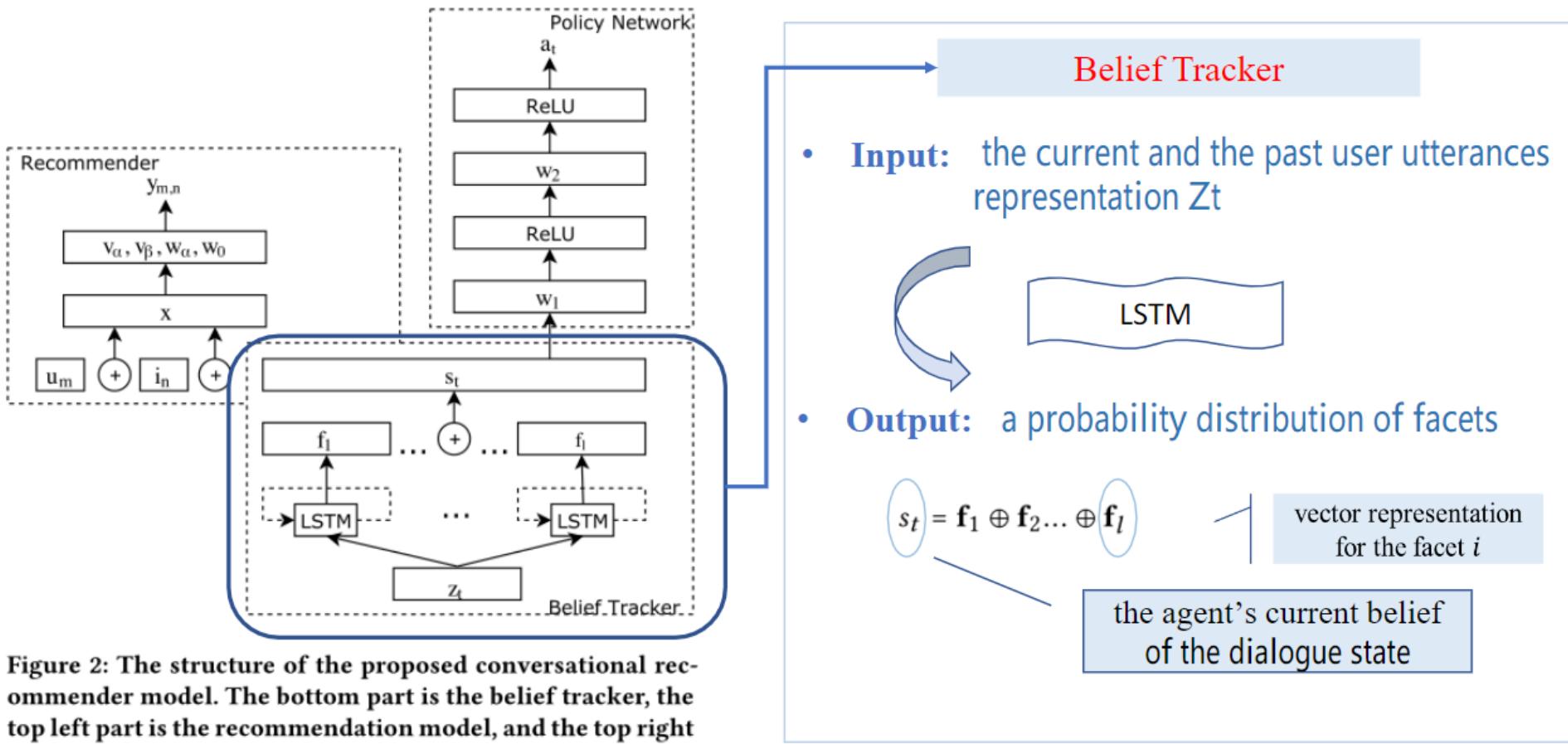


Figure 2: The structure of the proposed conversational recommender model. The bottom part is the belief tracker, the top left part is the recommendation model, and the top right part is the deep policy network.

Belief Tracker:

semi-structured query \rightarrow (attribute, metadata) / (color, red)

$$\mathbf{z}_t = ngram(e_t)$$

$$\mathbf{h}_t = LSTM(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_t)$$

$$\mathbf{f}_i = softmax(\mathbf{h}_t)$$

$$s_t = \mathbf{f}_1 \oplus \mathbf{f}_2 \dots \oplus \mathbf{f}_l$$

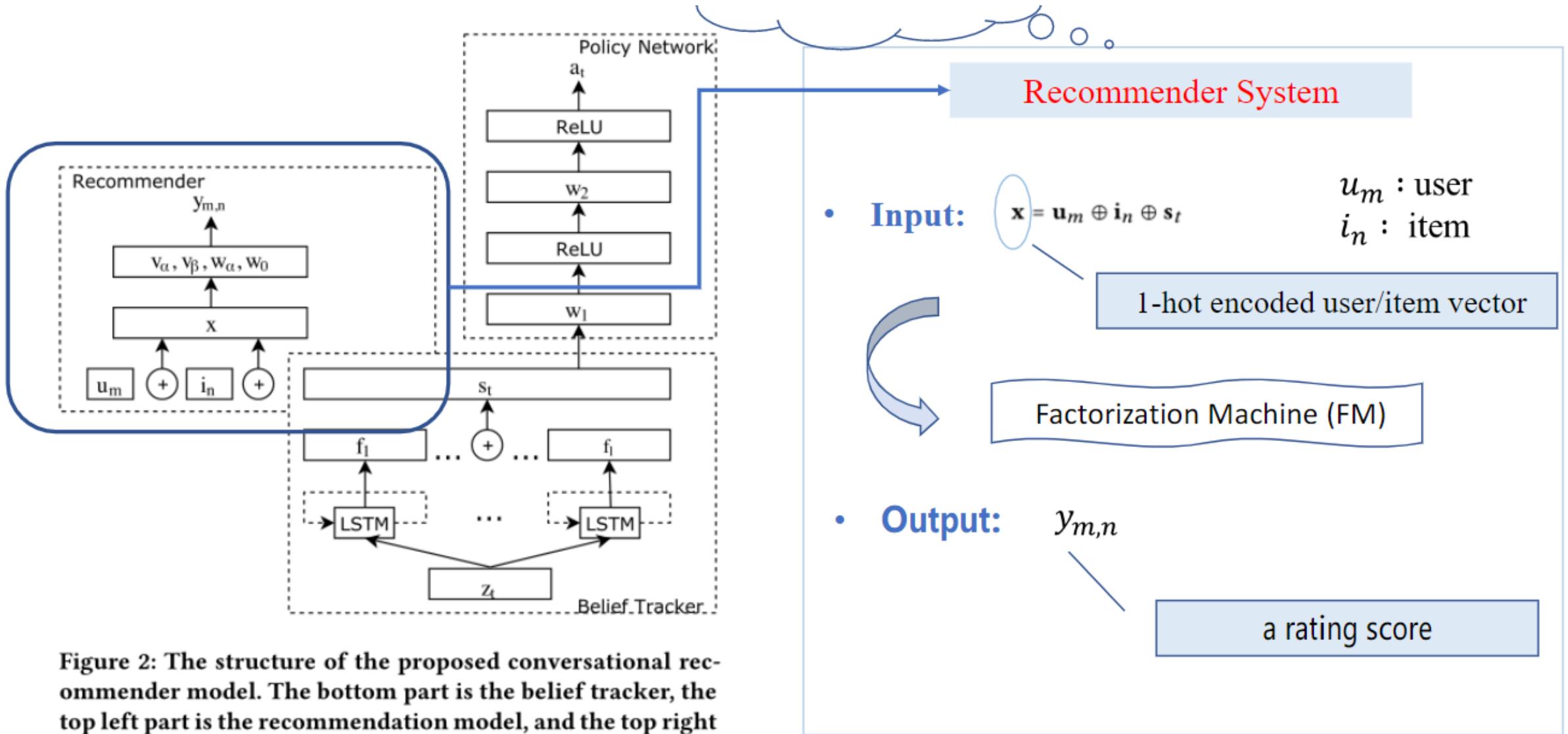


Figure 2: The structure of the proposed conversational recommender model. The bottom part is the belief tracker, the top left part is the recommendation model, and the top right part is the deep policy network.

• CRM - Method

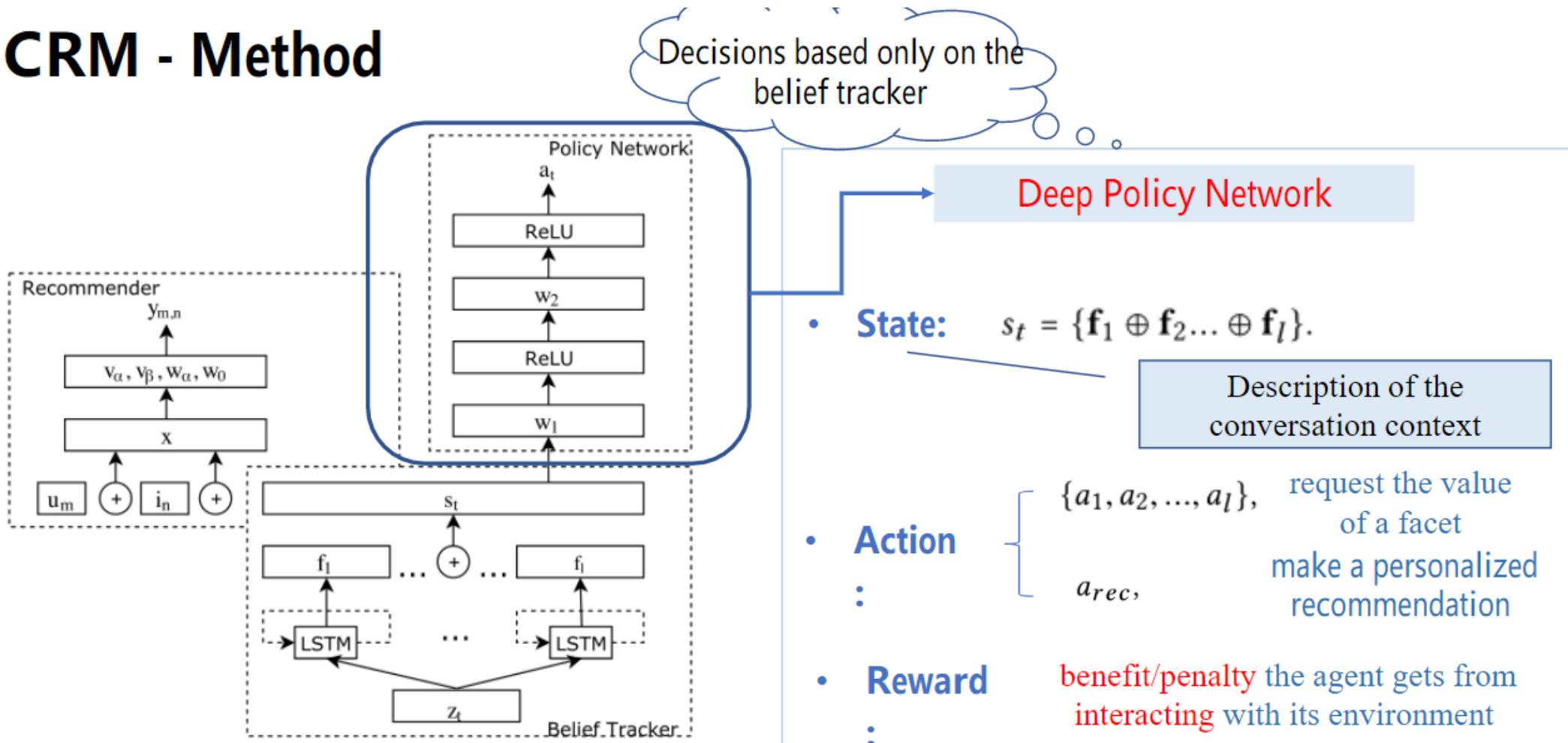


Figure 2: The structure of the proposed conversational recommender model. The bottom part is the belief tracker, the top left part is the recommendation model, and the top right part is the deep policy network.

User Simulation

Table 2: The schema based dialogue collection example. The bold utterances are written by the crowd sourcing workers via rewriting the templates to interact with the agent.

The target restaurant has the following facets.

{category: Mexican, state: AZ, city: Glendale,
price range: cheap, rating range: >=3.5}

User: inform(city="Glendale", category="Mexican")

User Write: I'm looking for Mexican food in Glendale.

Agent: Which state are you in?

User: inform(state="AZ")

User Write: I'm in Arizona.

Agent: Which price range do you like?

User: inform(price_range="cheap")

User Write: Low price.

Agent: What rating range do you want?

User: inform(rating_range>="3.5")

User Write: 3.5 or higher

Agent: <make recommendations>

User: thanks()

User Write: thank you

Behavior:

- (1) answering the agent's question. the value of a new facet.
- (2) finding the target item in the recommendation list.
success/fails
- (3) leaving the dialogue. dialogue is too long

Numerical reward:

- (1) r_q : a negative reward when the user quits the conversation
- (2) r_p : positive reward when the target in the recommendations
- (3) r_c : small negative reward per dialogue turn

(city="Italian", category="San Diego")

I'm looking for Italian food in San Diego.

Which state are you in?

I'm in California. (state="CA")

Which price range do you like?

Low price (price_range="cheap")

Algorithm 1 The interaction between agent and simulated user

```
1: Start with M epochs, N training data
2: for epoch = 1, M do
3:   for i = 1, N do
4:     t = 0
5:     Sample a  $(u, i)$  pair from the training set
6:     The user  $u$  starts the conversation, and conveys a random facet value with utterance  $e_t$ 
7:     while True do
8:       Apply belief tracker to  $e_t$  to get  $s_t$ 
9:       Send  $s_t$  to the policy gradient agent
10:      Get action  $a_t$  from the agent
11:      if  $a_t$  is  $a_{rec}$  then
12:        Call the recommender to get an item ranking
list
13:        if the target item is in the top K then
14:           $r_t = r_p$ , the dialogue succeeds and break
15:        else
16:           $r_t = r_q$ , the dialogue fails and break
17:        else
18:          The system generates a response  $e_m$  to the user
19:          if the user quits then
20:             $r_t = r_q$ , the dialogue fails and break
21:          else
22:             $r_t = r_c$ , t = t + 1
23:            The user responds with a new utterance  $e_t$ 
```

Table 1: Basic data statistics

	Number of Values
Users	62047
Items	21350
User-item pairs	875721
Category	191
State	13
City	189
Price	4
Rating Range	9

Table 3: Comparisons of CRM and the baselines on R (the average success rate), T (average number of turns), S (the success rate), W (the wrong quit rate) and L (the low rank rate).

Methods	R	T	S	W	L
MaxEnt@1	-9.308	1.0	2.28	2.64	95.08
MaxEnt@2	-6.496	1.523	12.60	5.44	81.96
MaxEnt@3	10.683	2.550	60.84	8.64	30.52
MaxEnt@4	20.648	3.492	82.08	8.60	9.32
MaxEnt Full	20.670	4.351	84.36	8.68	6.96
CRM	21.781	3.666	85.00	7.24	7.76

Estimation–Action–Reflection: Towards Deep Interaction Between Conversational and Recommender Systems

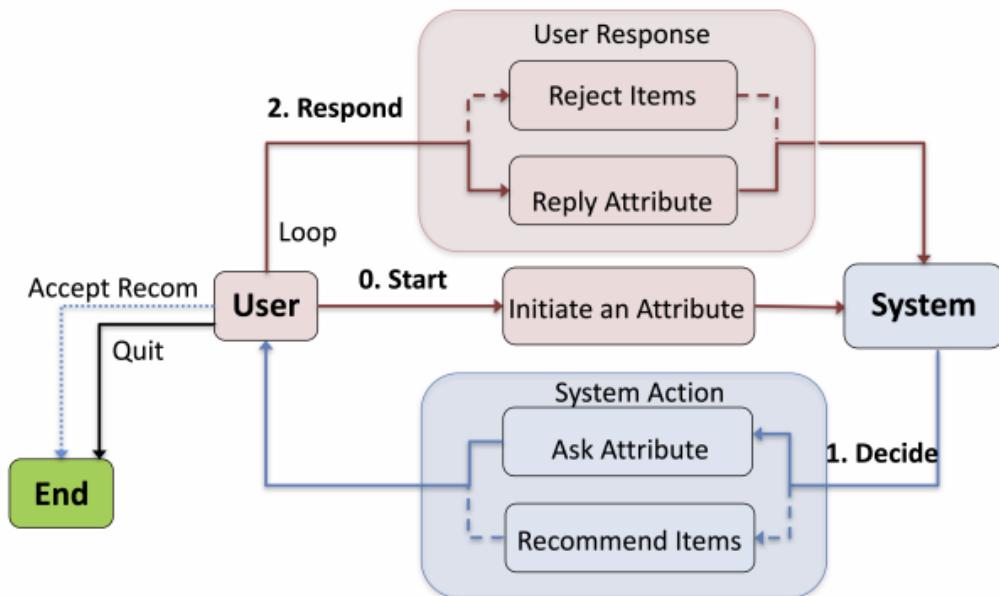


Figure 1: The workflow of our multi-round conversational recommendation scenario. The system may recommend items multiple times, and the conversation ends only if the user accepts the recommendation or chooses to quit.

- **Estimation:** estimate user preference on items and attributes;
- **Action:** learns a dialogue policy to determine whether to ask or recommend, what attribute to ask.
- **Reflection:** updates the recommender model when a user rejects the recommendations made by the Action stage.
- conversation scenarios on **binary** and **enumerated** questions

conversational component (**CC**)

responsible for interacting with the user

recommender component (**RC**)

responsible for estimating user preference (attribute/item)

• EAR - Method -- What Item to Recommend and What Attribute to Ask

Method: Attribute-aware FM for Item Prediction and Attribute Preference Prediction

Notation	Meaning
p	A given attribute
\mathbf{u}	User embedding
\mathcal{P}_u	User's known preferred attributes

Notation	Meaning
(Neg. 1) $\mathcal{V}_u^- := \mathcal{V} \setminus \mathcal{V}_u^+$	The ordinary negative sample as in standard BPR.
(Neg. 2) $\widehat{\mathcal{V}}_u^- := \mathcal{V}_{cand} \setminus \mathcal{V}_u^+$	\mathcal{V}_{cand} is the set of candidate items satisfying user's preferred attributes.
$\mathcal{D}_1 := \{(\mathbf{u}, \mathbf{v}, \mathbf{v}') \mathbf{v}' \in \mathcal{V}_u^-\}$	Paired sample for first kind of negative sample
$\mathcal{D}_2 := \{(\mathbf{u}, \mathbf{v}, \mathbf{v}') \mathbf{v}' \in \widehat{\mathcal{V}}_u^-\}$	Paired sample for second kind of negative sample

$$\hat{y}(\mathbf{u}, \mathbf{v}, \mathcal{P}_u) = \mathbf{u}^T \mathbf{v} + \sum_{p_i \in \mathcal{P}_u} \mathbf{v}^T \mathbf{p}_i$$

Score function for item prediction

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

ordinary negative example

The items satisfying the specified attribute but still are not clicked by the user

• EAR - Method -- What Item to Recommend and What Attribute to Ask

Method: Attribute-aware FM for Item Prediction and Attribute Preference Prediction

Notation	Meaning
p	A given attribute
u	User embedding
\mathcal{P}_u	User's known preferred attributes

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

Score function for attribute preference prediction

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

Notation	Meaning
(Neg. 1) $\mathcal{V}_u^- := \mathcal{V} \setminus \mathcal{V}_u^+$	The ordinary negative sample as in standard BPR.
(Neg. 2) $\widehat{\mathcal{V}}_u^- := \mathcal{V}_{cand} \setminus \mathcal{V}_u^+$	\mathcal{V}_{cand} is the set of candidate items satisfying user's preferred attributes.
$\mathcal{D}_1 := \{(u, v, v') v' \in \mathcal{V}_u^-\}$	Paired sample for first kind of negative sample
$\mathcal{D}_2 := \{(u, v, v') v' \in \widehat{\mathcal{V}}_u^-\}$	Paired sample for second kind of negative sample

$$L = L_{item} + L_{attr}$$

Multi-task Learning: Optimize for item ranking and attribute ranking simultaneously.

• EAR - Method -- Action stage

Method: Strategy to Ask and Recommend? (Action Stage)

We use **reinforcement learning** to find the best strategy. $\pi(a^t | s^t)$

- policy gradient method
- simple policy network (2-layer feedforward network)

• **State Vector** $s = s_{ent} \oplus s_{pre} \oplus s_{his} \oplus s_{len}$

• **$s_{entropy}$: The entropy of attribute is important.**

• **$s_{preference}$: User's preference on each attribute.**

• **$s_{history}$: Conversation history is important.**

• **s_{length} : Candidate item list length.**

Reward $r_t = r_{suc} + r_{ask} + r_{quit} + r_{prev}$

$r_{success}$: Give the agent a big reward when it successfully recommend!

r_{ask} : Give the agent a small reward when it ask a correct attribute.

r_{quit} : Give the agent a big negative reward when the user quit (the conversation is too long)

Note: 3 of the 4 information come from Recommender Part

Action Space: $|\mathcal{P}| + 1$ $\mathcal{A} = \{a_{rec} \cup \{a_{ask}(p) | p \in \mathcal{P}\}\}$

$r_{prevent}$: Give each turn a relatively small reward to prevent the conversation goes too long.

• EAR - Method -- Reflection

Method: How to Adapt to User's Online Feedback? (Reflection stage)

Solution: We treat the recently rejected 10 items as negative samples to re-train the recommender, to adjust the estimation of user preference.

$$L_{ref} = \sum_{(u,v,v') \in \mathcal{D}_4} -\ln \sigma(\hat{y}(u, v, \mathcal{P}_u) - \hat{y}(u, v', \mathcal{P}_u)) + \lambda_{\Theta} \|\Theta\|^2$$

Notation	Meaning
\mathcal{V}^t	Recently rejected item set.
$\mathcal{D}_4 := \{(u, v, v') v' \in \mathcal{V}_u^+ \wedge v' \in \mathcal{V}^t\}$	Paired sample for online update.

• EAR - Evaluation

Table 1: Dataset statistics.

Dataset	#users	#items	#interactions	#attributes
Yelp	27,675	70,311	1,368,606	590
LastFM	1,801	7,432	76,693	33

Item Name: “*Small Italy Restaurant*”
Item Attributes: [Pizza, Nightlife, Wine, Jazz]



I'd like some Italian food.

Yes!

Check, I don't want
“Small Paris”

Got you, do you like some pizza?

Yes!

Got you, do you like some nightlife?

Rejected!

Do you want “Small Paris”?

Check, I don't want
“Rock Music”

Got you, do you like some Rock Music?

No!

Do you want “Small Italy Restaurant”?

Accepted!



Template-based
utterances

Results

Table 2: SR@15 and AT of compared methods. * denotes that improvement of EAR over other methods is statistically significant for $p < 0.01$ (RQ1).

	LastFM		Yelp	
	SR@15	AT	SR@15	AT
Abs Greedy	0.209	13.63	0.271	12.26
Max Entropy	0.290	13.61	0.919	5.77
CRM	0.325	13.43	0.923	5.33
EAR	0.429*	12.45*	0.971*	4.71*

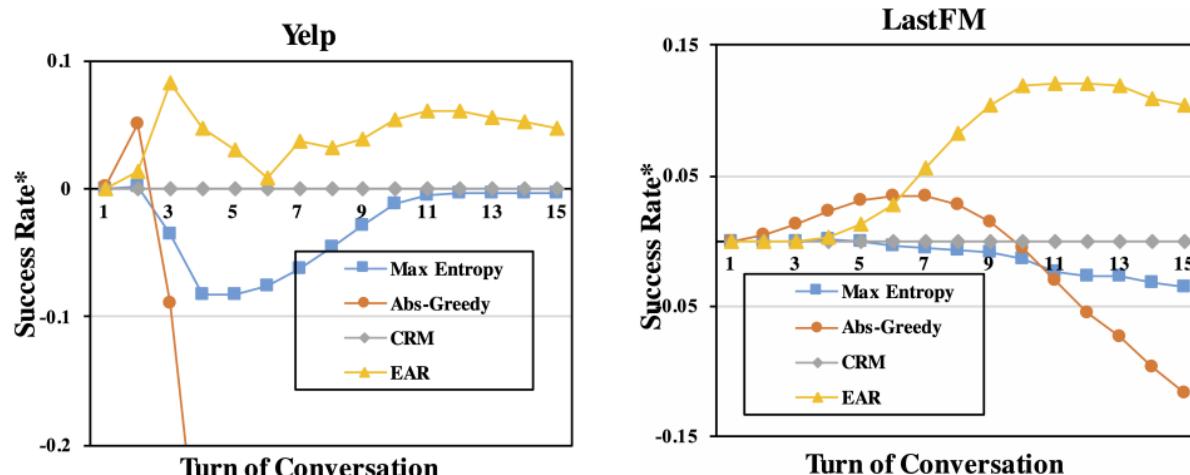


Figure 2: Success Rate* of compared methods at different conversation turns on Yelp and LastFM (RQ1).

Table 5: Performance after removing the online update module in the reflection stage. * denotes that improvement of EAR over removing update module is statistically significant for $p < 0.01$ (RQ4).

	Yelp				LastFM			
	SR@5	SR@10	SR@15	AT	SR@5	SR@10	SR@15	AT
-update	0.629	0.905	0.970	4.72	0.020	0.217	0.393	12.67
EAR	0.629	0.907	0.971	4.71	0.020	0.243*	0.429*	12.45*

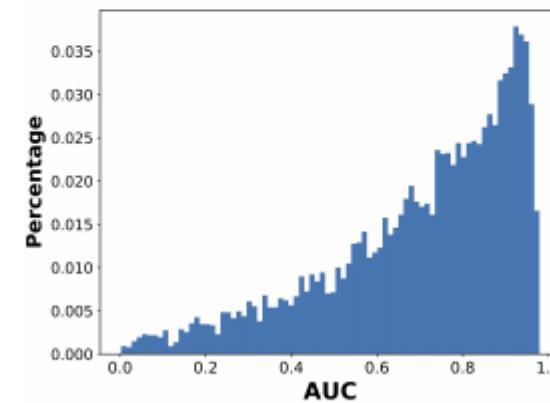
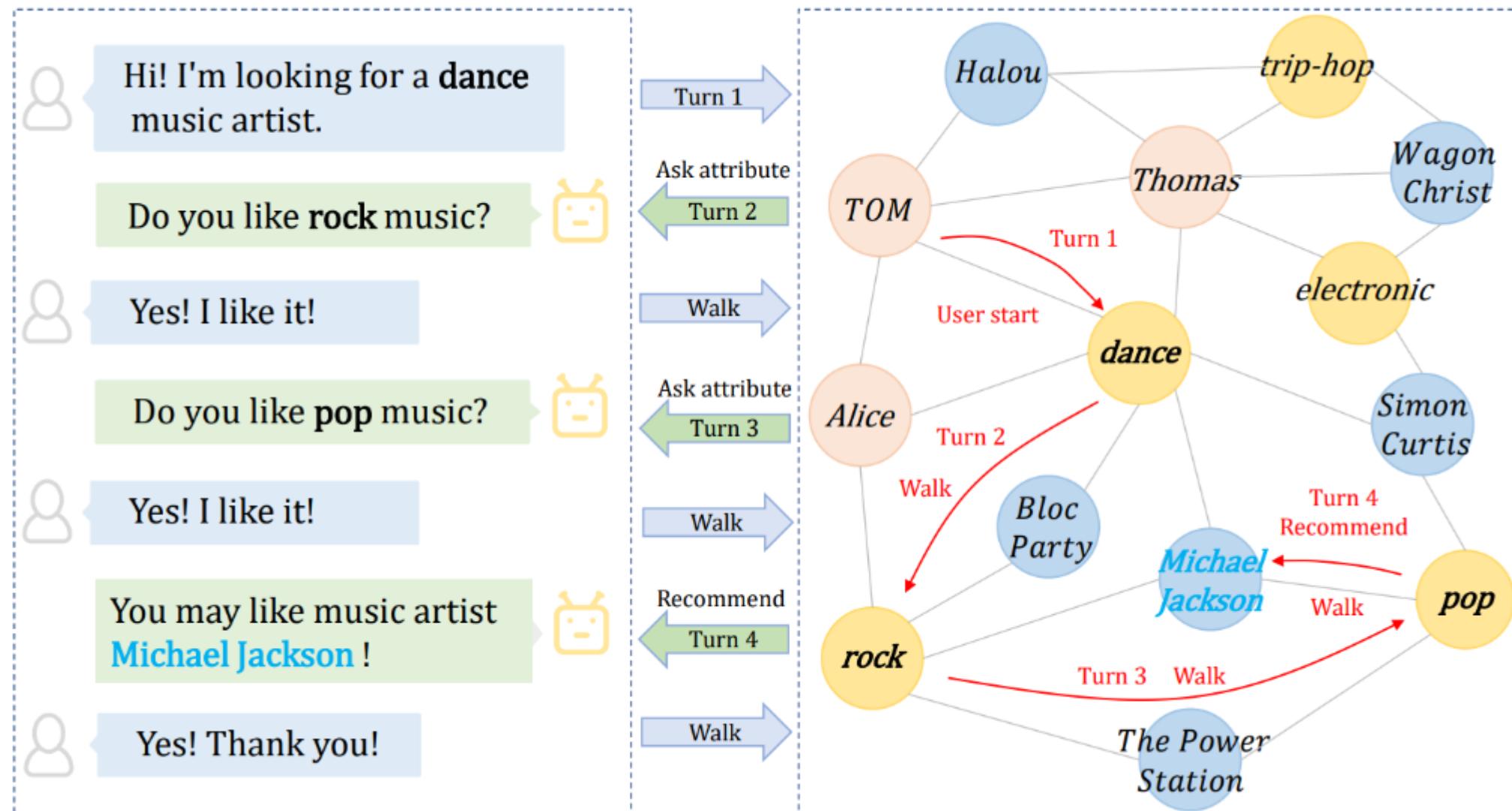


Figure 3: Percentage of bad updates w.r.t. the offline model's AUC on the users on Yelp (RQ4).

CPR - Motivation



• CPR - Method

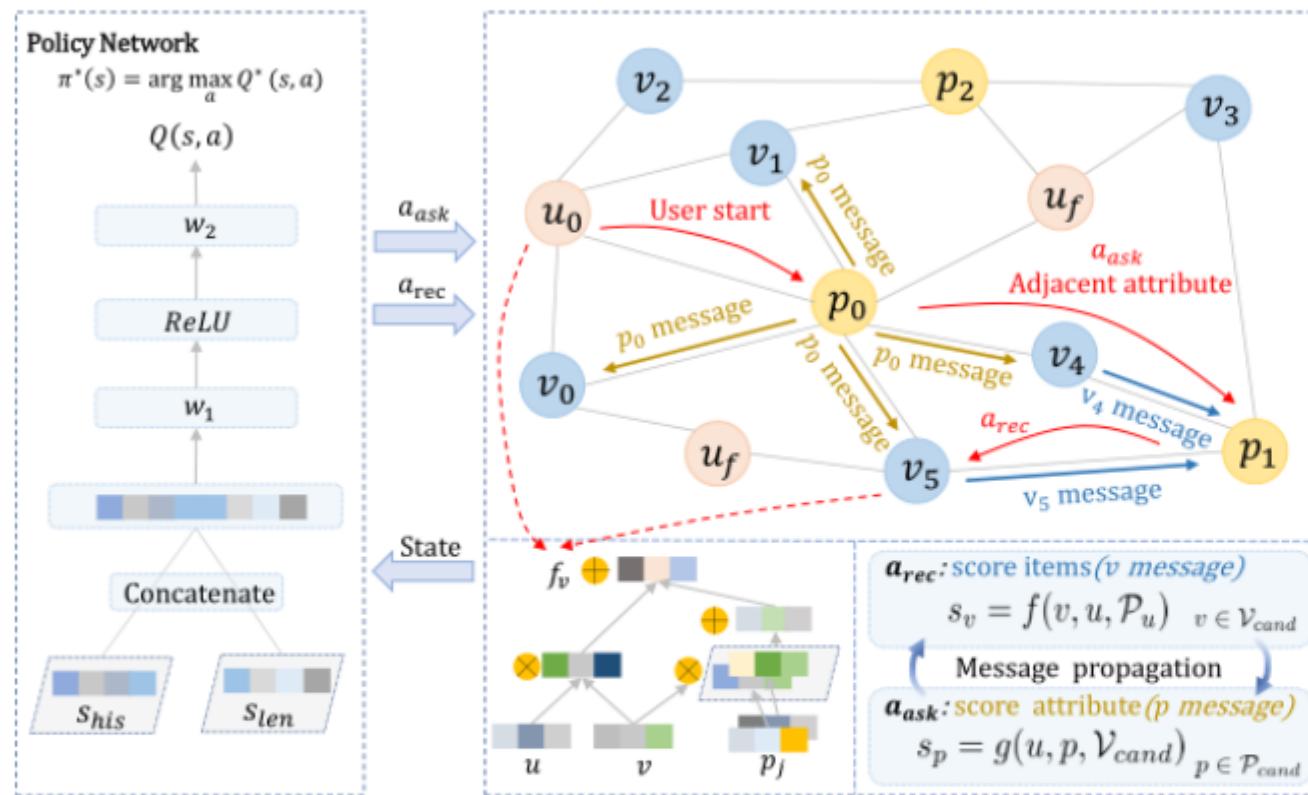


Figure 2: CPR framework overview. It starts from the user u_0 and walks over adjacent attributes, forming a path (the red arrows) and eventually leading to the desired item. The policy network (left side) determines whether to ask an attribute or recommend items in a turn. Two reasoning functions f and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

CPR Framework

• Assuming

- Current path $P = p_0, p_1, p_2, \dots, p_t$
- u : user v : item p : attribute
- \mathcal{P}_u : user's preferred attributes
- \mathcal{V}_{cand} : candidate items

• Reasoning

- Score items to recommend (v message):

$$s_v = f(v, u, \mathcal{P}_u) \quad v \in \mathcal{V}_{cand}$$
- Score attribute to ask (p message):

$$s_p = g(u, p, \mathcal{V}_{cand}) \quad p \in \mathcal{P}_{cand}$$

• Consultation

- Policy network (choose to ask or rec)

• Transition

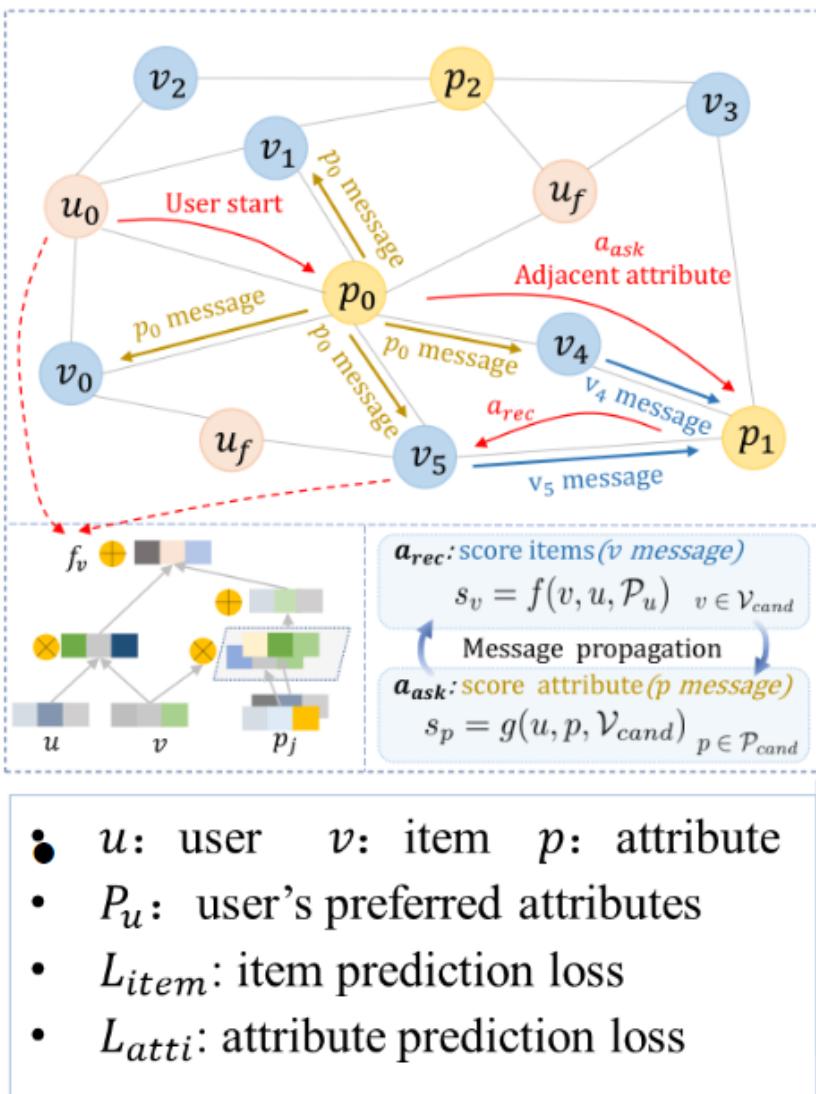
- Extended path

$$P = p_0, p_1, p_2, \dots, p_t, p_{t+1}$$

- Update candidate item /attribute set ($\mathcal{V}_{cand}/\mathcal{P}_{cand}$)

• CPR - Method

An instantiation of CPR Framework



Message propagation from attributes to items

Factorization Machine in EAR

- Item prediction

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

- Optimization:
Bayesian Personalized Ranking

The same with the recommender model in EAR

Message propagation from items to attributes

Information entropy strategy

- Weighted attribute information entropy

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

$$-\text{prob}(p) \cdot \log_2(\text{prob}(p)),$$

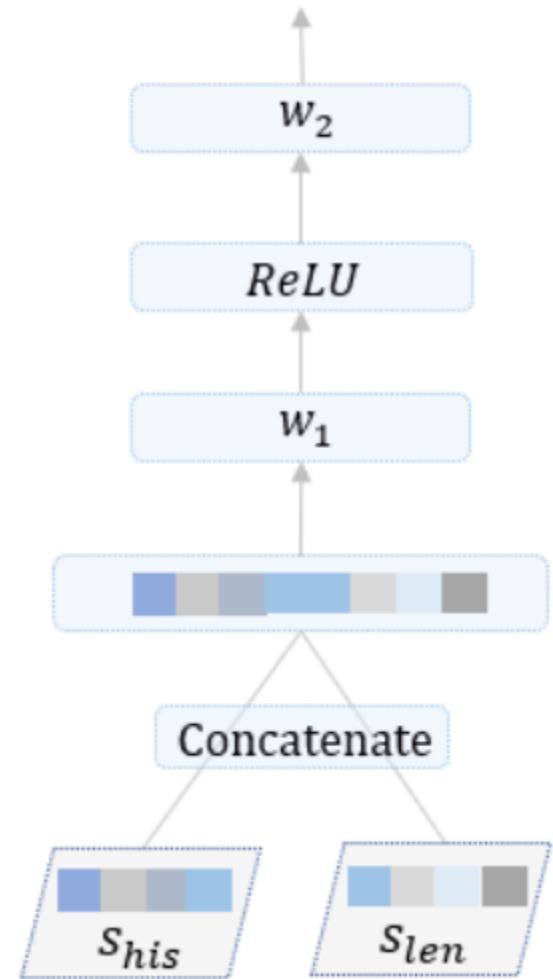
$$\text{prob}(p) = \frac{\sum_{v \in \mathcal{V}_{cand} \cap \mathcal{V}_p} \sigma(s_v)}{\sum_{v \in \mathcal{V}_{cand}} \sigma(s_v)}$$

• CPR - Method

Policy Network

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

$$Q(s, a)$$



Input $s = s_{his} \oplus s_{len}$

s_{his} : encodes the conversation history

s_{len} : encodes the size of candidate items

Output

$$Q(s, a)$$

$Q(s, a)$: the value of action a in state s

a_{rec} : the action of recommendation

a_{ask} : the action of asking attribute

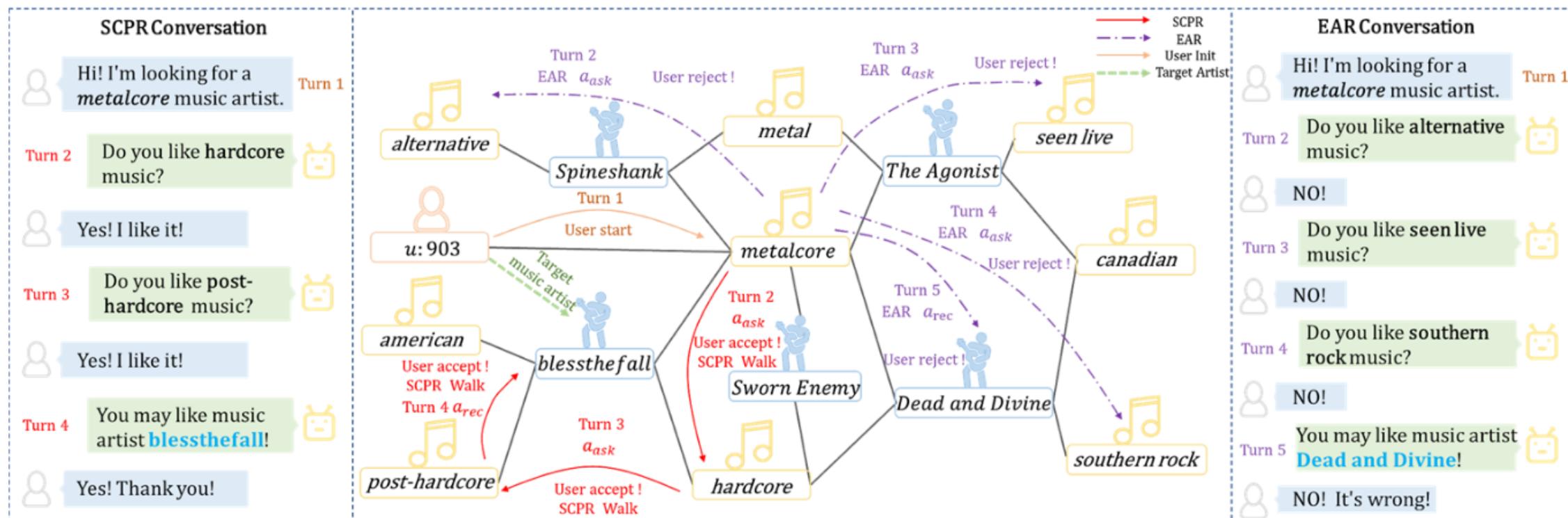
DQN method

Policy: $\pi^*(s) = \arg \max_a Q^*(s, a)$

TD loss: $\delta = Q(s, a) - \left(R + \gamma \max_a Q(s', a) \right)$

• CPR - Evaluation

CPR can make the reasoning process explainable and easy-to-interpret!



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

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

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

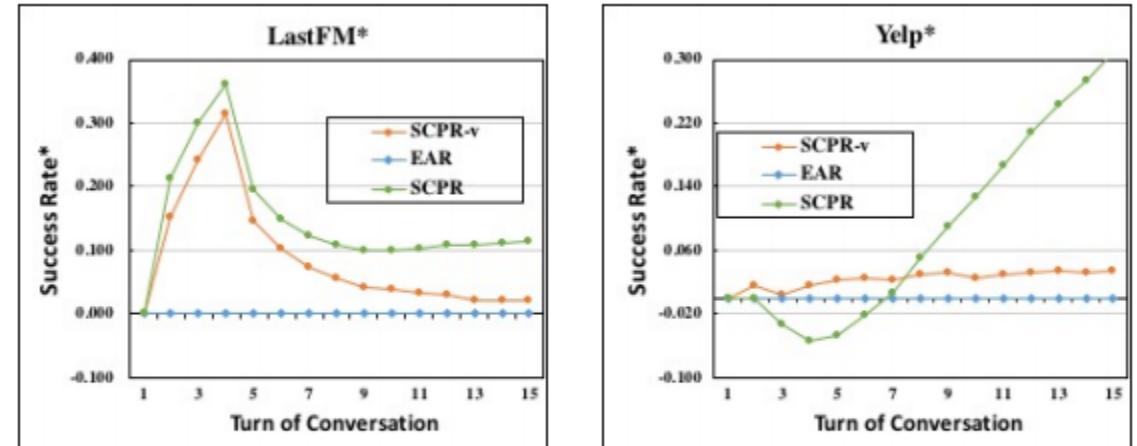


Figure 6: Success Rate* of compared methods at different conversation turns on LastFM* and Yelp*(RQ2).