Estimation–Action–Reflection:

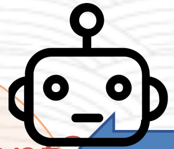
Towards Deep Interaction Between Conversational and Recommender Systems

Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, Tat-Seng Chua
{wenqianglei, xiangnanhe, miaoyisong}@gmail. com, qw2ky@virginia. edu, hongrc@hfut. edu. cn,
{kanmy,chuats}@comp. nus. edu. sg

NUS National University of Singapore School of Computing
University of Science and Technology of China 1958
VIRGINIA
合肥工業大學 HEFEI UNIVERSITY OF TECHNOLOGY

What is conversational recommendation

want a new phone.



What operating system do you want?

Asking attribute

iOS

What about the latest iPhone 11?

Attempt to recommend

Reflect on why user reject recommended items?

No, too expensive.

Do you want all screen design with FaceID?

Asking attribute

Yes!

Do you want more color options? Red, blue?

Asking attribute

Red is great option

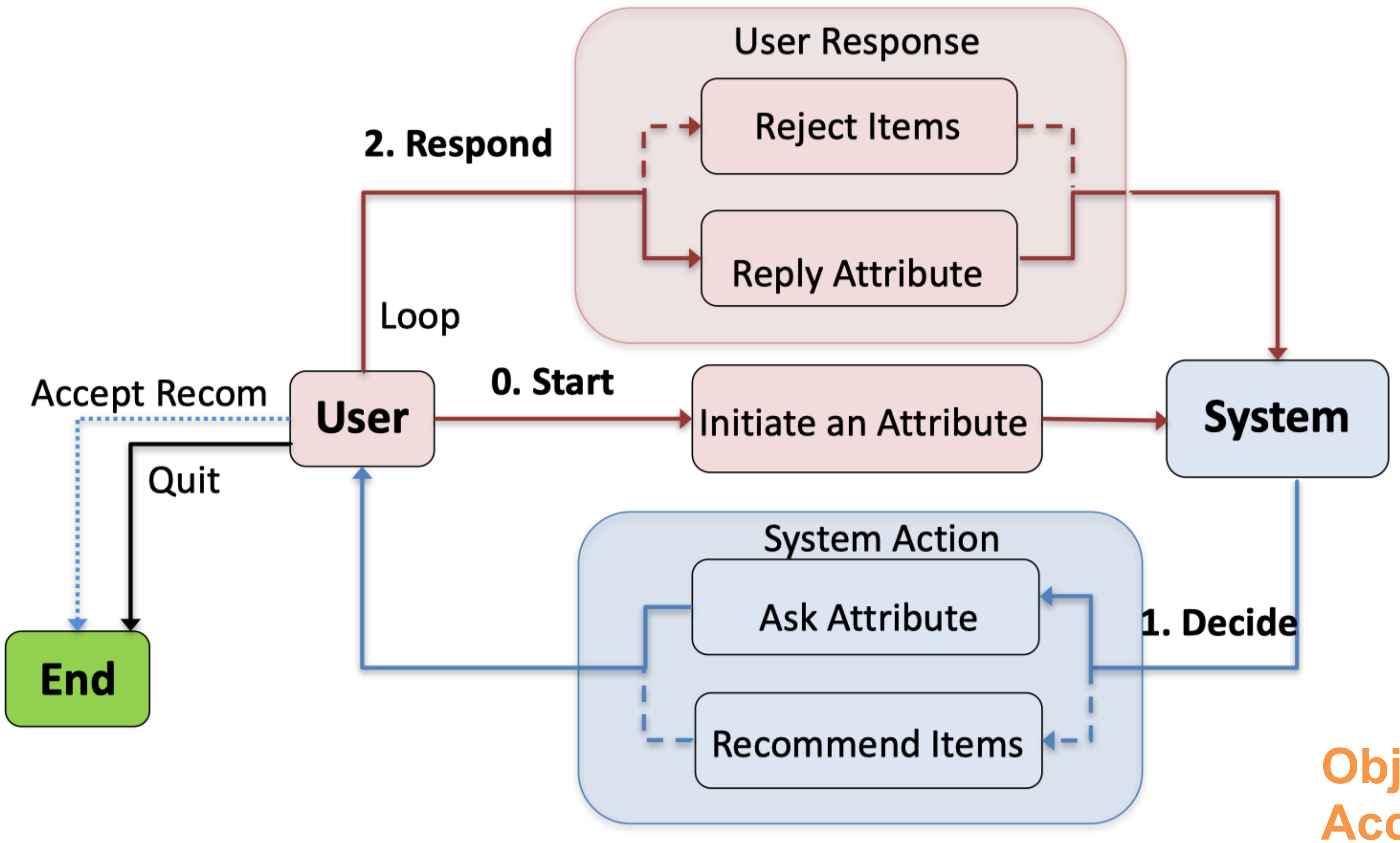
iPhone XR Red with 128GB is a real bargain!

Attempt to recommend

User accept, conversation terminates.

Nice! I will take it!

Workflow of multi-round Conversational Recommendation Scenario



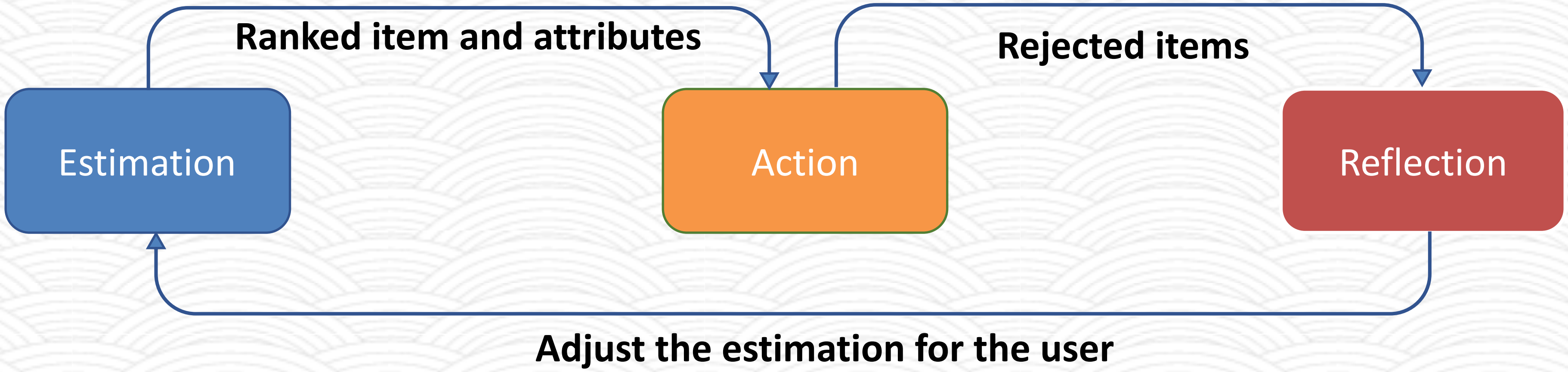
Our proposed multi-round scenario

- One session is started by the user specifying a desired attribute.
- One session will be stopped only when the recommendation is successful or the user quits.

Objective:
Accurately recommend item to user in shortest turns

Method: EAR- Estimation, Action, Reflection

Deep interaction among CC(conversation system) and RC (recommendation system)



Estimation:

- RC ranks the candidate item and item attribute.

Action:

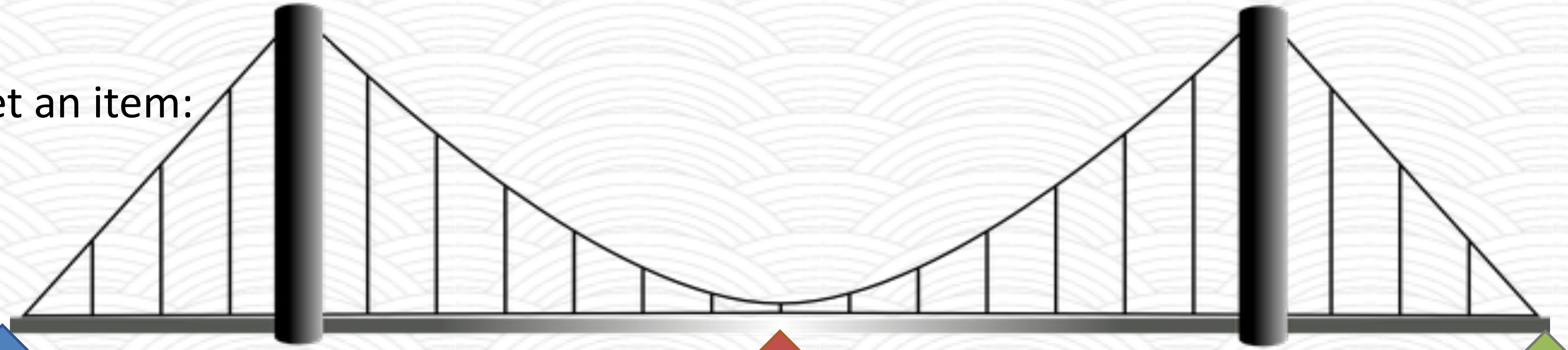
- CC takes into account ranked items and ranked attributes to decide whether to ask attribute or make recommendation

Reflection:

- When user rejects list of recommendation, the RC adjusts its estimation for user.

The Position of Conversational Recommendation—Bridging Recommendation System and Search

Traditional method for user to get an item:
Search or **Recommendation**



Search:
User's Intention is totally clear

Conversational Recommendation:
Try to induce user preference through conversation!

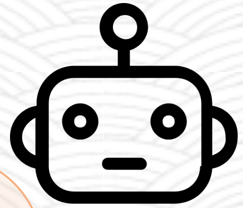
Recommendation:
User's Intention is totally unclear

- We have 3 Key Research Tasks:

  1. **What item to recommend? What attribute to ask?**
  2. **Strategy to ask and recommend?**
  3. **How to adapt to user's online feedback?**

Objective:
Accurately recommend item to use
in shortest turns

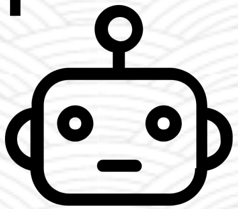
Estimation stage — Item prediction



1000 candidates remains	I'd like some **Italian** food.
	Got you, do you like some **Pizza**?
250 candidates remains	Yes!
	Got you, do you also want some **nightlife**?
95 candidates remains	Yes!

- How to rank top that restaurant she really wants within all candidates remained?

Estimation stage — Attribute prediction



1000 candidates remains

I'd like some **Italian** food.

Got you, do you like some **Chinese food**?

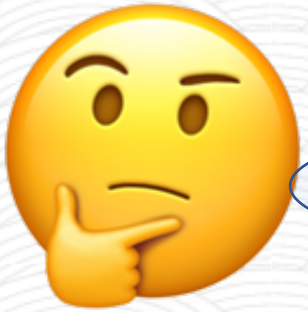
1000 candidates remains. Waste a turn!

No!

Got you, do you also want some __**?**__?

__?__ candidates remains

___?___



- What question should I ask next, so she can give me positive feedback? given the attributes I already know.

Preliminary - FM (Factorization Machine) De Facto Choice for recommender system

Feature vector **x** | Target y

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$x^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$x^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$x^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$x^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$x^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$x^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

- A framework to learn embedding in a same vector space.
- Capture the interaction between vectors by their inner product.
- Co-occur, similar.

Notation	Meaning
u	User embedding
v	Item embedding
P_u={p_1,p_2,…, p_n}	Known user preferred attributes in current conversation session.

Score Function to decide how likely user would like an item:

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

Method: Bayesian Personalized Ranking

$$L_{bpr} = \sum_{(u,v,v')\in\mathcal{D}_1} -\ln\sigma\left(\hat{y}(u,v,\mathcal{P}_u) - \hat{y}(u,v',\mathcal{P}_u)\right) + \lambda_\Theta\|\Theta\|^2$$

Positive sample

Negative sample

Notation	Meaning
$\mathcal{D}_1 := \{(u,v,v') \mid v' \in \mathcal{V}_u^-\},$	Paired sample for BPR learning
v	The positive sample: the item in current conversation
$\mathcal{V}_u^- := \mathcal{V} \setminus \mathcal{V}_u^+$	Item that user never interacted with
σ	Sigmoid function
λ_Θ	Regularization

Method: Attribute-aware BRP for item prediction and attribute preference prediction

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.
$\mathcal{D}_1 := \{(u, v, v') \mid v' \in \mathcal{V}_u^-\}$	Paired sample for first kind of negative sample
$\mathcal{D}_2 := \{(u, v, v') \mid v' \in \widehat{\mathcal{V}_u^-}\}$	Paired sample for second kind of negative sample

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

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

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

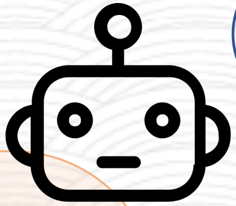
$$\hat{g}(p \mid u, \mathcal{P}_u) = \mathbf{u}^T p + \sum_{p_i \in \mathcal{P}_u} \mathbf{P}^T \mathbf{P}_i$$

Score function for attribute preference prediction

$$L = L_{item} + L_{attr}$$ Multi-task Learning

Note: We use information gathered by CC(conversation part) to enhance the RC!

Action stage: Strategy to ask and recommend?



This time, I try to recommend more earlier...

1000 candidates remains	I'd like some **Italian** food.
	Got you, do you like some **pizza**?
250 candidates remains	Yes!
	Got you, do you like some **nightlife**?
95candidates remains	Yes!
	Try to recommend 10 items! ← Should recommend?
95candidates remains	**Rejected!**
	Got you, do you like some **Wine**?
30 candidates remains	Yes!
	Try to recommend 10 items! ← Should recommend?
Target item rank 6 / 10	**Accepted!**

Method: Strategy to ask and recommend? (Action Stage)

We use **reinforcement learning** to find the best strategy.

- policy gradient method
- simple policy network of 2-layer feedforward network

- **State Vector**
- $s_{entropy}$: **The entropy of attribute is important.**
- $s_{prefrence}$: **User's preference on each attribute.**
- $s_{history}$: **Conversation history is important.**
- s_{length}: **Candidate item list length.**

Note: 3 of the 4 information come from Recommender Part

Action Space: $|\mathcal{P}| + 1$

Reward

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

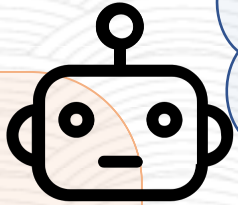
r_{ask}: Give the agent a small reward when it ask a correct attribute.

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

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

Reflection stage: How to adapt to user's online feedback?

This time, I try to recommend more earlier...



1000 candidates remains

250 candidates remains

95candidates remains

Adjust estimation

I'd like some Italian food.

Got you, do you like some pizza?

Yes!

Got you, do you like some nightlife?

Yes!

Try to recommend 10 items!

Should recommend?

Rejected!

She rejected my recommended 10 items... However, that is what she should love according to her history. How can I induce her current preference with this 10 items?

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.

$$\boldsymbol{L_{ref}} = \sum_{(\boldsymbol{u},\boldsymbol{v},\boldsymbol{v}')\in\mathcal{D}_4} -\ln\sigma\left(\widehat{\boldsymbol{y}}(\boldsymbol{u},\boldsymbol{v},\mathcal{P}_{\boldsymbol{u}}) - \widehat{\boldsymbol{y}}(\boldsymbol{u},\boldsymbol{v}',\mathcal{P}_{\boldsymbol{u}})\right) + \boldsymbol{\lambda}_{\Theta}\|\boldsymbol{\Theta}\|^2$$

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

Experiment setup (I) - Dataset Collection

Dataset Description

Dataset	#user	#item	#interactions	#attributes
Yelp	27,675	70,311	1,368,606	590
Last.FM	1,801	7,432	76,693	33

Why we need to create dataset?

- There's no existing datasets specially for CRS as this field is very new.
- Datasets of previous work has too few attributes for real-world applications.

How we create dataset?

- Standard pruning operation (user / item has < 5 reviews)
- For Last.FM, we build 33 Binary attributes for Last.FM (Classic, Popular, Rock, etc...)
- For Yelp, we build 29 enumerated attributes on a 2-level taxonomy over 590 original attributes.

Experiment setup (2)

User simulator

- Lack an offline experiment environment for conversational recommendation.
- We use the real interactions pair between user and item.
- The user simulator will keep the target item in “its heart”, then give responses interactively to our agents. Responses include give answer to a question, and accept/reject item when our agent proposes a list of recommendation.

Training details

- We set the max length of conversation to 15, and fix the length of recommendation list to 10.
- We use SGD optimizer to train FM model(hidden size = 64), with L2 regularization of 0.001, the learning rate of item prediction is 0.01 and attribute prediction is 0.001
- For the policy network(MLP), we use 2 layer hidden size of 256, we pre-train it as a classifier according to max-entropy results, and use REINFORCE algorithm to train with learning rate of 0.001. r_success = 1, r_ask=0.1, r_quit=-0.3, r_prevent=-0.1, discount factor γ=0.7

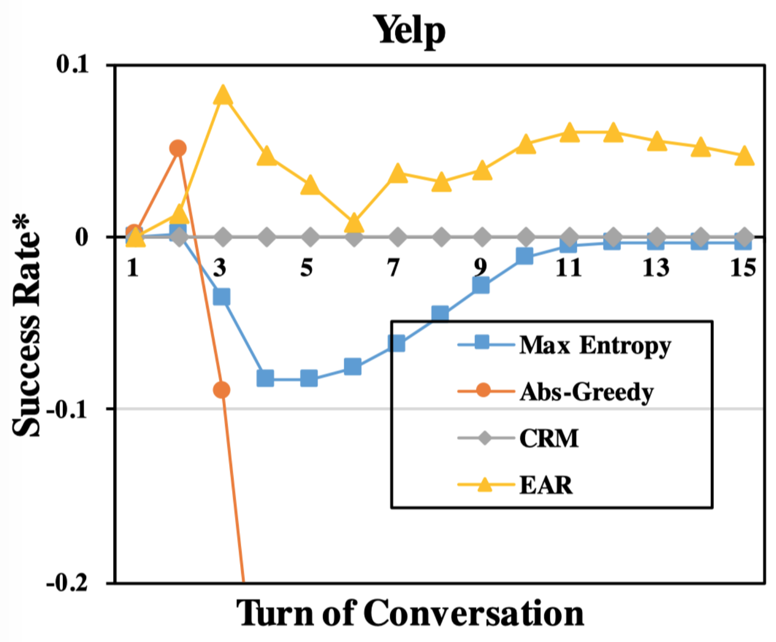
Main Experiment Results

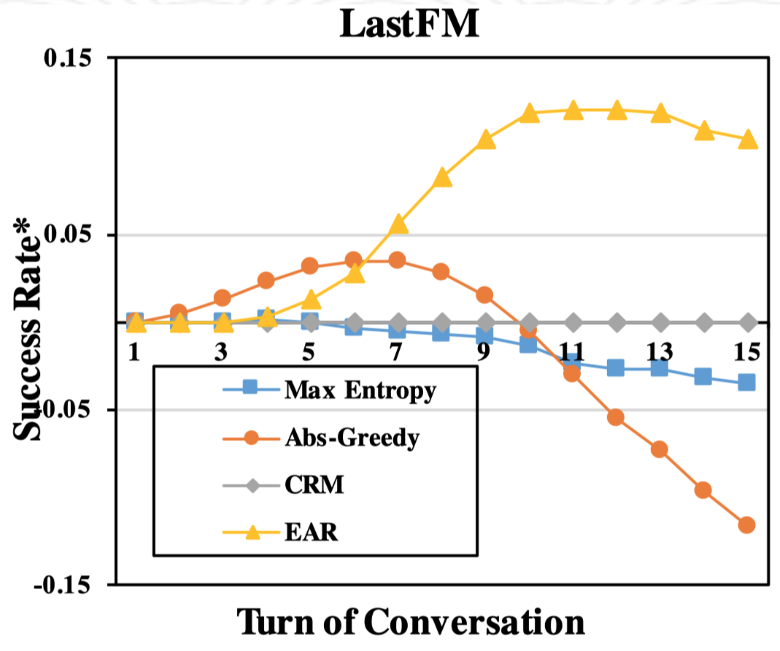
Evaluation Matrices:

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

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***





Experiment results – Estimation stage

item and attribute prediction

	LastFM		Yelp	
	Item	Attribute	Item	Attribute
FM	0.521	0.727	0.834	0.654
FM+A	0.724	0.629	0.866	0.638
FM+A+MT	**0.742***	**0.760***	**0.870***	**0.896***

The offline AUC score of prediction of item and attributes

- Standard FM model,
- FM + A (attribute aware item BPR)
- FM + A + MT (Multitask learning)

Experiment results – Action stage

Strategy to ask and recommend?

Table 4: Performance of removing one component of the state vector (Equation 10) from our EAR. * denotes that improvement of EAR over model with removed component is statistically significant for $p < 0.01$ (RQ 3).

	Yelp				LastFM			
	SR@5	SR@10	SR@15	AT	SR@5	SR@10	SR@15	AT
$-\mathbf{s}_{ent}$	0.614	0.895	0.969	4.81	**0.051**	0.190	0.346	12.82
$-\mathbf{s}_{pre}$	0.596	0.857	0.959	5.06	0.024	0.231	0.407	12.55
$-\mathbf{s}_{his}$	0.624	0.894	0.949	4.79	0.021	0.236	0.424	12.50
$-\mathbf{s}_{len}$	0.550	0.846	0.952	5.44	0.013	0.230	0.416	12.56
EAR	**0.629***	**0.907***	**0.971***	**4.71***	0.020	**0.243***	**0.429***	**12.45***

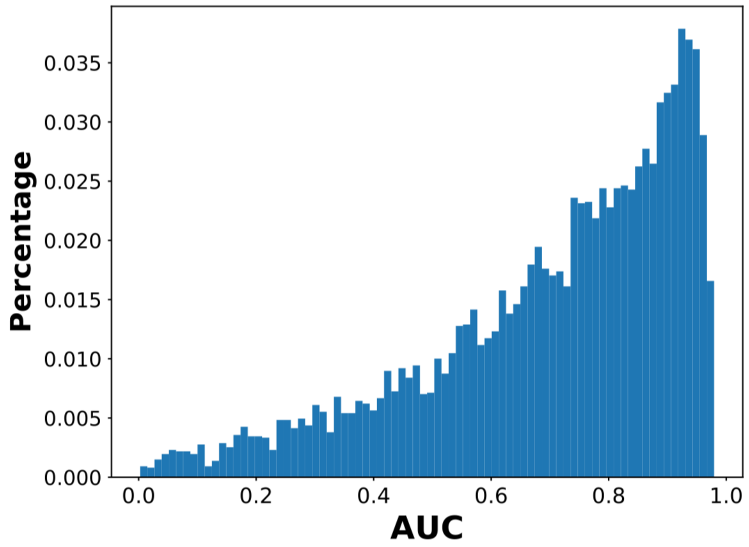
We conducted ablation study on the state vector fed into policy network, in order to find the contribution of each component.

- entropy seems to be the most salient component.

Experiment Result: Reflection stage
How to adapt to user's online feedback?

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***



"Bad update" in Yelp Dataset

Performance of removing the online update module. Yelp suffers less than LastFM, Why?

- Yelp dataset has a better offline AUC.
- When offline AUC is higher, the reflection stage tend to have less effect.

Conclusion and Future Works

- We formalize the task of multi-turn conversational recommendation
- We refine the recommendation system in a conversational scenario for attribute-aware item ranking and attribute-aware preference estimation.
- We proposes a three-stage solution EAR for CRS, outperforming state-of-the-art baselines.
- We plan to do online evaluation and obtain real-world exposure data by collaborating with E-commerce companies.

Thank you!

Spare Slides

Importance of this research project

The Importance of CRS (Conversational Recommendation System):

- Overcome the limitation of traditional static recommender systems, thus improve user's satisfaction and bring revenue for business!
- Embrace recent advances in conversation technology.

The Advances Brought By Our Work:

- We're the first to consider a realistic multi-round conversational recommendation scenario.
- Unifying CC(Conversation Component) and RC(Recommender Component), and propose a novel three-staged solution EAR.
- We build two datasets by simulating user conversations to make the task suitable for offline academic research.

Literature Review (1)

- **Static Traditional Recommendation Systems:**

- Collaborative Filtering
- Matrix Factorization
- Factorization Machine
- etc...

- **Limitation 1:**

- Offline: learn from user history data, so can only mimic user's history preference.

- **Limitation 2:**

- User cannot explicit tell system her preference.
- System cannot leverage user's feedback.

Existing online recommendation methods (bandit):

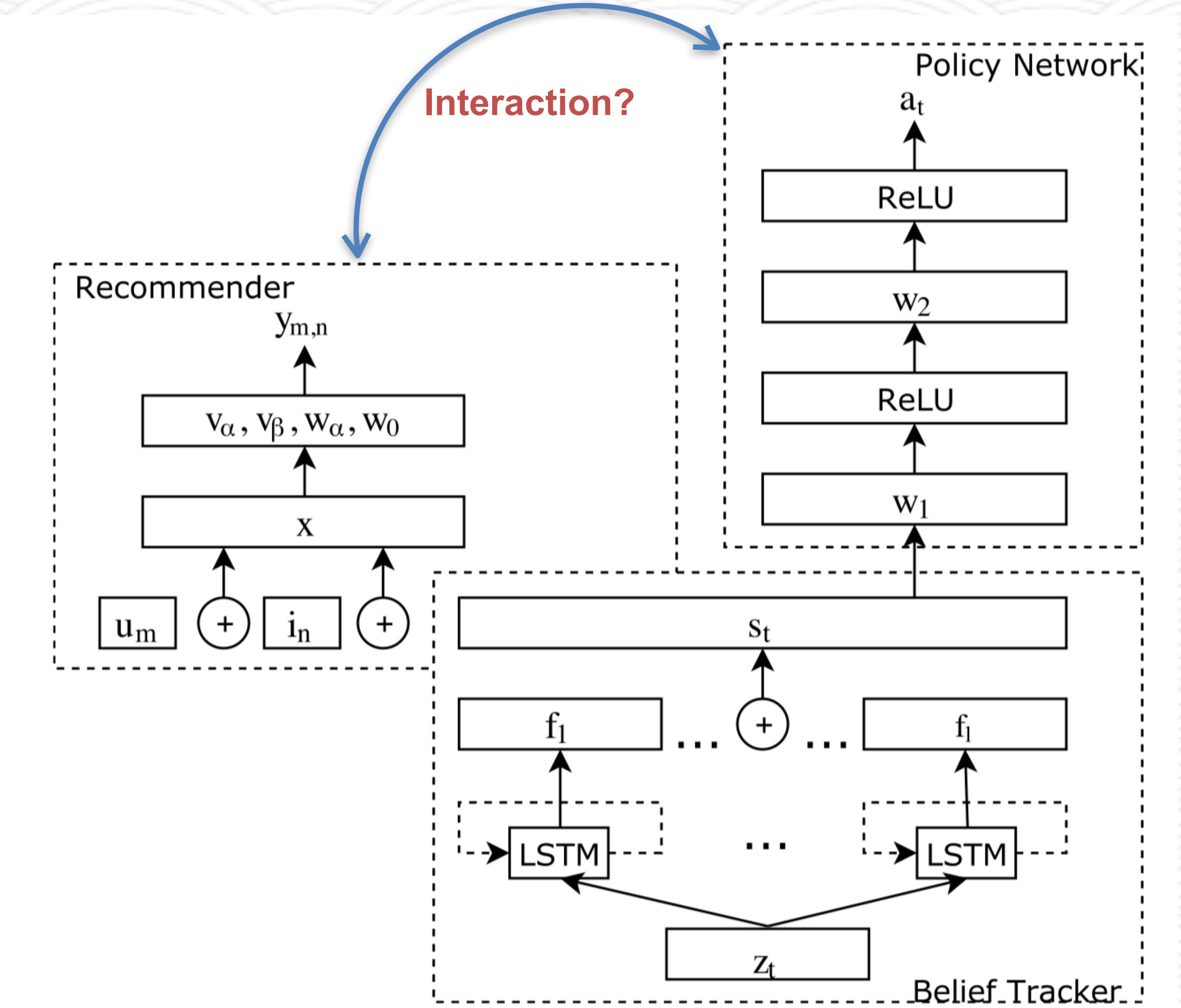
- epsilon-greedy
- Thompson-Sampling
- Upper Confidence Bound (UCB)
- Linear-UCB
- Collaborative UCB...

Limitation:

- Can only attempt to recommend items, cannot ask attributes of item
- The mathematics formulation of bandit restricts it to only recommend 1 item each turn.

Literature Review (2)

Towards Conversational Recommendation — Sun et.al. SIGIR 2018



Screenshot from SIGIR 2018, Towards Conversational Recommendation

Limitation:

- Can only recommend for 1 time. The session will end regardless of success or not.
- Recommender Component and Conversation Component are isolated part.
- Simply taking belief tacker as input for action decision.