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
I want a new phone.

What operating system do you want?

iOS

What about the latest iPhone 11?

No, too expensive.

Do you want all screen design with FaceID?

Yes!

Do you want more color options? Red, blue?

Red is great option

iPhone XR Red with 128GB is a real bargain!

Nice! I will take it!

Reflect on why user reject recommended items?

User accept, conversation terminates.
Workflow of multi-round Conversational Recommendation 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\textsubscript{(conversation system)} and RC\textsubscript{(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
- Recommendation: User's Intention is totally unclear
- Conversational Recommendation: Try to induce user preference through conversation!

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 user in shortest turns
Estimation stage — Item prediction

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

- How to rank top that restaurant she really wants within all candidates remained?
Estimation stage — Attribute prediction

I’d like some **Italian** food.

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

No!

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

___?

• 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

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

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

\[
\hat{y}(u, v, P_u) = u^T v + \sum_{p_i \in P_u} v^T p_i
\]

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>User embedding</td>
</tr>
<tr>
<td>v</td>
<td>Item embedding</td>
</tr>
<tr>
<td>P_u={p_1,p_2,..., p_n}</td>
<td>Known user preferred attributes in current conversation session.</td>
</tr>
</tbody>
</table>
**Method:** Bayesian Personalized Ranking

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

- **Positive sample**
- **Negative sample**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{D}_1 := {(u, v, v')</td>
<td>v' \in \mathcal{V}_u^-} )</td>
</tr>
<tr>
<td>( v )</td>
<td>The positive sample: the item in current conversation</td>
</tr>
<tr>
<td>( \mathcal{V}_u^- := \mathcal{V} \setminus \mathcal{V}_u^+ )</td>
<td>Item that user never interacted with</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>Sigmoid function</td>
</tr>
<tr>
<td>( \lambda_{\theta} )</td>
<td>Regularization</td>
</tr>
</tbody>
</table>
**Method**: Attribute-aware BRP for item prediction and attribute preference prediction

- **Notation**: 
  
  \[ \mathcal{V}_u := \mathcal{V} \setminus \mathcal{V}_u^+ \]  
  The ordinary negative sample as in standard BPR.

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

Score function for attribute preference prediction:

\[
L_{\text{item}} = \sum_{(u,v,v') \in \mathcal{D}_1} - \ln \sigma (\hat{y}(u,v,\mathcal{P}_u) - \hat{y}(u,v',\mathcal{P}_u)) + \sum_{(u,v,v') \in \mathcal{D}_2} - \ln \sigma (\hat{y}(u,v,\mathcal{P}_u) - \hat{y}(u,v',\mathcal{P}_u)) + \lambda_\theta \|\theta\|^2
\]

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

\[
\hat{g}(p|u,\mathcal{P}_u) = u^T p + \sum_{p_i \in \mathcal{P}_u} P_i^T p
\]

\[
L = L_{\text{item}} + L_{\text{attr}} \quad \text{Multi-task Learning}
\]
Action stage: Strategy to ask and recommend?

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!**

Rejected!

- Got you, do you like some **Wine**?
  - Yes!
- **Try to recommend 10 items!**

Accepted!

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

1000 candidates remains

250 candidates remains

95 candidates remains

95 candidates remains

30 candidates remains

Target item rank 6 / 10

Should recommend?

Should recommend?
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  
• $s_{\text{entropy}}$: The entropy of attribute is important.  
• $s_{\text{preference}}$: User's preference on each attribute.  
• $s_{\text{history}}$: Conversation history is important.  
• $s_{\text{length}}$: Candidate item list length.

Note: 3 of the 4 information come from Recommender Part

Reward  
$r_{\text{success}}$: Give the agent a big reward when it successfully recommend!  
$r_{\text{ask}}$: Give the agent a small reward when it ask a correct attribute.  
$r_{\text{quit}}$: Give the agent a big negative reward when the user quit (the conversation is too long)  
$r_{\text{prevent}}$: Give each turn a relatively small reward to prevent the conversation goes too long.

Action Space: $|\mathcal{P}| + 1$
Reflection stage: How to adapt to user's online feedback?

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!

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?

1000 candidates remains

250 candidates remains

95 candidates remains

Adjust estimation

Should recommend?
**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.

\[ L_{ref} = \sum_{(u,v,v') \in D_4} - \ln \sigma (\hat{y}(u,v,P_u) - \hat{y}(u,v',P_u)) + \lambda_\theta \| \Theta \|^2 \]

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{V}^t )</td>
<td>Recently rejected item set.</td>
</tr>
<tr>
<td>( D_4 := {(u,v,v')</td>
<td>v' \in \mathcal{V}_u^+ \land v' \in \mathcal{V}^t} )</td>
</tr>
</tbody>
</table>
Experiment setup (1) - Dataset Collection

Dataset Description

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#user</th>
<th>#item</th>
<th>#interactions</th>
<th>#attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yelp</td>
<td>27,675</td>
<td>70,311</td>
<td>1,368,606</td>
<td>590</td>
</tr>
<tr>
<td>Last.FM</td>
<td>1,801</td>
<td>7,432</td>
<td>76,693</td>
<td>33</td>
</tr>
</tbody>
</table>

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 $\gamma=0.7$
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).

<table>
<thead>
<tr>
<th></th>
<th>LastFM</th>
<th>Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR@15</td>
<td>AT</td>
</tr>
<tr>
<td>Abs Greedy</td>
<td>0.209</td>
<td>13.63</td>
</tr>
<tr>
<td>Max Entropy</td>
<td>0.290</td>
<td>13.61</td>
</tr>
<tr>
<td>CRM</td>
<td>0.325</td>
<td>13.43</td>
</tr>
<tr>
<td>EAR</td>
<td>0.429*</td>
<td>12.45*</td>
</tr>
</tbody>
</table>
### Experiment results – Estimation stage:

**item and attribute prediction**

<table>
<thead>
<tr>
<th></th>
<th>LastFM</th>
<th></th>
<th>Yelp</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Item</td>
<td>Attribute</td>
<td>Item</td>
<td>Attribute</td>
</tr>
<tr>
<td>FM</td>
<td>0.521</td>
<td>0.727</td>
<td>0.834</td>
<td>0.654</td>
</tr>
<tr>
<td>FM+A</td>
<td>0.724</td>
<td>0.629</td>
<td>0.866</td>
<td>0.638</td>
</tr>
<tr>
<td>FM+A+MT</td>
<td><strong>0.742</strong></td>
<td><strong>0.760</strong></td>
<td><strong>0.870</strong></td>
<td><strong>0.896</strong></td>
</tr>
</tbody>
</table>

The offline AUC score of prediction of item and attributes:

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

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).

<table>
<thead>
<tr>
<th></th>
<th>Yelp</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR@5</td>
<td>SR@10</td>
<td>SR@15</td>
<td>AT</td>
<td>SR@5</td>
<td>SR@10</td>
<td>SR@15</td>
<td>AT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-s_{ent}$</td>
<td>0.614</td>
<td>0.895</td>
<td>0.969</td>
<td>4.81</td>
<td><strong>0.051</strong></td>
<td>0.190</td>
<td>0.346</td>
<td>12.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-s_{pre}$</td>
<td>0.596</td>
<td>0.857</td>
<td>0.959</td>
<td>5.06</td>
<td>0.024</td>
<td>0.231</td>
<td>0.407</td>
<td>12.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-s_{his}$</td>
<td>0.624</td>
<td>0.894</td>
<td>0.949</td>
<td>4.79</td>
<td>0.021</td>
<td>0.236</td>
<td>0.424</td>
<td>12.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-s_{len}$</td>
<td>0.550</td>
<td>0.846</td>
<td>0.952</td>
<td>5.44</td>
<td>0.013</td>
<td>0.230</td>
<td>0.416</td>
<td>12.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAR</td>
<td><strong>0.629</strong></td>
<td><strong>0.907</strong></td>
<td><strong>0.971</strong></td>
<td><strong>4.71</strong></td>
<td><strong>0.020</strong></td>
<td><strong>0.243</strong></td>
<td><strong>0.429</strong></td>
<td><strong>12.45</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th></th>
<th>Yelp</th>
<th></th>
<th>LastFM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR@5</td>
<td>SR@10</td>
<td>SR@15</td>
<td>AT</td>
</tr>
<tr>
<td>update</td>
<td>0.629</td>
<td>0.905</td>
<td>0.970</td>
<td>4.72</td>
</tr>
<tr>
<td></td>
<td>0.020</td>
<td>0.217</td>
<td>0.393</td>
<td>12.67</td>
</tr>
<tr>
<td>EAR</td>
<td>0.629</td>
<td>0.907</td>
<td>0.971</td>
<td>4.71</td>
</tr>
<tr>
<td></td>
<td>0.020</td>
<td>0.243*</td>
<td>0.429*</td>
<td>12.45*</td>
</tr>
</tbody>
</table>

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 propose 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.
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.