Towards Learning User-Adaptive State Models in a Conversational Recommender System

Tariq Mahmood[1] and Francesco Ricci[2]

[1]University of Trento, Trento, Italy

tariq@itc.it

[2]Free University of Bozen-Bolzano, Bolzano, Italy

fricci@unibz.it

Contents

- Background and Motivation
  - **Conversational Recommendation Models** and their **limitations**
  - Our proposed **Adaptive Recommendation Model**
  - System considers information (encoded as features of a state representation) in order to learn adaptive behaviour
  - *Crucial Task*: Determine the *relevant* state features for a given recommendation task

- Investigate State Feature Relevancy
  - *Evaluation Setup*: Different (simulated) user models and different state representations
  - Relevancy Criteria
- Results and Current Work

Conversational Recommendation Models

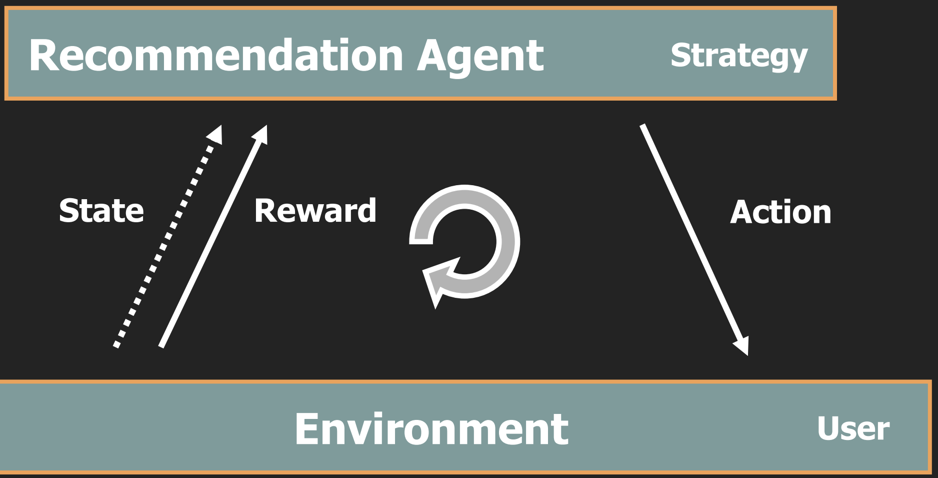
- At each **stage** of the conversation, the system executes **one action** from amongst a set of alternative actions e.g. query the user, show most popular products etc.
- User's feedback is used to adapt the **user model** and hence, the future recommendations
- They follow some *strategy* or plan of action during their interaction session with the user
- Limitations: They are not able to learn the strategy by themselves
  - The strategy is always pre-determined by the system designers and hard-coded inside the system in advance, e.g., critique-based or preference-based.

Adaptive Recommendation Model

- At **each stage**, our system *autonomously decides* to execute the action that it believes:
  - the user is more likely to accept rather than reject
  - is more likely to bring the user towards her goal *in the long run*, i.e. at the end of the interaction session
  - As experiences (sessions) are collected, the system eventually learns the most adaptive action at each stage, i.e., it *improves its current strategy in order to learn an optimal strategy*
- Basically, the system acquires some information (encoded as features of a state representation) at each stage in order to learn the optimal behaviour.

RL cycle

- Learn by interacting with an environment, through the consequences of actions rather than through explicit teaching (Sutton and Barto, 98).



$$s_0 \xrightarrow{a_0 : r_0} s_1 \xrightarrow{a_1 : r_1} s_2 \xrightarrow{a_2 : r_2} \cdots$$

RL Concepts

- A discounted infinite-horizon reward model

  γ is the discount parameter. $0 \le \gamma \le 1$, R_T is the total interaction reward

$$\boxed{R_T = \sum_{t=0}^{\infty} \gamma^t R_t}$$

- The policy π followed by the agent is a function that assigns an action to each state. We use the term optimal policy instead of optimal strategy in the context of the agent's actions.

$$\boxed{\pi : S \rightarrow A}$$

Concepts…

- The value function of a policy π $V^{\pi}(s)$ that gives for each state $s \in S$ the expected sum of reward obtained starting from s and following π thereafter:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s'), \forall s \in S$$

- The agent adopts the optimal policy π^* when:
  - For each state $s \in S$, if the agent behaves according to π^*, then the expected total reward $V^*(s)$ is a maximum

$$\pi^*(s) = arg \max_a (R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s'))$$

Query Tightening Example

- Current policy: When a query retrieves too many items the system suggests features to the user for *tightening* the query
- Is this an optimal policy?
- Maybe it is better not to use query tightening at all.

Results

- State Representation: {Page-User Action (PUA), Current Result Size (CRS), Expected Result Size (ERS)}
- For each value of cost, the agent is always able to improve the initial policy in order to adopt an optimal one [Ricci and Mahmood, 2007].

Action	Description
Suggest	Suggest Tightening Features
Execute	Execute Query and Show Results

Table 1: System Action Set

$Value$	$Situation$
$+1$	user adds a product to her cart
-0.05	an interaction session stage elapses

Table 2: Reward Function

	Optimal Policy Actions for states with $pua = qf - eq$					
$cost$	(s, s)	(m, s)	(m, m)	(l, s)	(l, m)	(l, l)
NP	exec	exec	exec	sugg	sugg	sugg
-0.01	exec	exec	exec	exec	exec	exec
-0.02	exec	exec	exec	exec	exec	sugg
-0.04	exec	exec	sugg	exec	sugg	sugg
-0.08	exec	sugg	sugg	sugg	sugg	sugg
-0.12	sugg	sugg	sugg	sugg	sugg	sugg

s : small m : medium l : large $sugg$: suggest $exec$: execute

Feature Relevancy

- Generally, **a lot of potentially useful information** could be considered in the state representation
  - system activity, user activity, session activity
  - considering all information is computationally infeasible for RL techniques
- For a given recommendation task, the system must be able to determine the *relevant* features from amongst an available set
- We attempt to determine feature relevancy for **different state representations**, under **different simulated user models**.
  - whether relevancy is influenced by the user behaviour.

Different State Representations

State Feature	Discretized Feature Values
Page-UserAction (PUA)	{*S-go, QF-execq, T-acct, T-rejt, T-modq R-modq, R-add, G*}
Current Result Size (CRS)	{*small (0 - 20) , medium (20 - 50) large (50 - 100) , verylarge (100 - INF)*}
Expected Result Size (ERS)	{*small (0 - 20) , medium (20 - 50) large (50 - 100) , verylarge (100 - INF)*}
Freq Tight Sugg (FTSugg)	{*small (0 - 2) , medium (2 - 4) , large (4 - INF)*}
Number Int Stages (NStages)	{*small (0 - 3), medium (3 - 6) , large (6 - INF)*}
User Tighten Resp (UserTResp)	{*accept, mixed, reject*}

Table 3: State Features (*INF*=Infinity)

- R = {R1,R2,R3,R4}

Rep	State Feature Set
Baseline	{*PUA, CRS*}
Rep1	{*PUA, CRS, ERS*}
Rep2	{*PUA, CRS, FTSugg*}
Rep3	{*PUA, CRS, NStages*}
Rep4	{*PUA, CRS, UserTResp*}

Different User Models

- The models differ based on how the simulated user responds to tightening suggestions
  - Generic UM (*GUM*)
  - Willing UM (*WillUM*)
  - Moderately-Willing UM (*ModwillUM*)
  - Un-willing UM (UnwillUM)
  - All UM (*AllUM*)
- 25 Optimal Policies.

Relevancy Criteria

- We propose two relevancy criteria for determining feature relevance:
  - *Policy Evaluation*
    - based on an *evaluation* of the optimal policies, i.e., on determining and comparing the total reward which the system can accumulate while taking actions according to a particular OP
    - average cumulative reward for 300 test items
  - *Policy Comparison*
    - based on a *comparison* of optimal policies
    - relevant if the new feature changes the policy of the corresponding old states.

Results – *Policy Evaluation*

UM	Different State Representations				
	Baseline	*Rep1*	*Rep2*	*Rep3*	*Rep4*
GUM	0.521	**0.5369**	**0.5623**	0.5213	**0.5491**
Will	0.5696	0.5749	0.5628	0.5576	0.5436
Mod	0.5326	0.5469	**0.5625**	**0.5633**	**0.5526**
Unwill	0.5636	0.5629	0.5491	0.5636	0.5539
All	0.5459	0.5399	**0.5626**	0.5437	0.5418

Table 5: Average cumulative rewards under different "User Model - State Representation" combinations (*UM*=user model, *Will* = *WillUM*, *Mod* = *ModwillUM*, *Unwill* = *UnwillUM*, *All* = *AllUM*)

- Rewards (for other representations) which are significantly larger than the reward for *Baseline* representation (according to a paired t-test) are marked in bold
- Results prove that the relevancy is influenced by the user behaviour
- Best to determine relevancy for a user population, i.e., under *AllUM*
  - *The best feature to add is fTSugg.*

Results – *Policy Comparison*

- Generally speaking, the results imply that, if more features are added to *Baseline*,
  - it is best to execute the query for smaller result sizes
  - the user population is not too willing to accept tightening even for large result sizes, and
  - it is best to suggest tightening only for very large result sizes
- Under *AllUM*, each representation in R is relevant, i.e., for our user population, it is better to add all our proposed features to *Baseline*
  - these results again prove that the relevancy is influenced by the user behavior.

Relevancy Criteria Comparison

- The results for both the criteria are different
  - Different criteria lead to different results
  - Need to standardize techniques for determining feature relevancy.

Current Work

- Adding a feature is not always relevant
- Best to consider behaviour for a user population
  - We have applied our recommendation model to an online travel recommender system
  - etPackaging project funded by the Austrian Network for E-Tourism
  - Currently we are running experiments in order to acquire data for learning the optimal policy for this system.

Related Work

- To the best of our knowledge, our work is the first attempt in addressing the relevancy problem in the domain of recommender systems
- [Tetreault and Litman, 2006] exploit the *Policy Comparison* criteria to prove the relevancy of five state representations under a corpus of real-user sessions
  - their results need further validation because we have shown that simply learning different actions doesn't guarantee that the new policy is optimal for the users.
- [Frampton and Lemon, 2006] adopt a similar criteria to *Policy Evaluation* in order to prove the relevancy of adding two dialogue features to a baseline representation.

References

- [Frampton and Lemon, 2006] Matthew Frampton and Oliver Lemon. Learning more effective dialogue strategies using limited dialogue move features. In ACL'06, 2006.
- [Ricci and Mahmood, 2007] Francesco Ricci and Tariq Mahmood. Learning and adaptivity in interactive recommender systems. In Proceedings of the ICEC'07 Conference, August 2007.
- [Tetreault and Litman, 2006] Joel R. Tetreault and Diane J. Litman. Using reinforcement learning to build a better model of dialogue state. In EACL, 2006.

Thank you!