Towards Sharing Task Environments to Support Reproducible Evaluations of Interactive Recommender Systems

Andrea Barraza-Urbina
Insight Centre for Data Analytics,
Data Science Institute, NUI Galway
andrea.barraza@insight-centre.org

Mathieu d'Aquin
Insight Centre for Data Analytics,
Data Science Institute, NUI Galway
mathieu.daquin@insight-centre.org

ABSTRACT

Beyond sharing datasets or simulations, we believe the Recommender Systems (RS) community should share *Task Environments*. In this work, we propose a high-level logical architecture that will help to reason about the core components of a RS Task Environment, identify the differences between Environments, datasets and simulations; and most importantly, understand what needs to be shared about Environments to achieve reproducible experiments. The work presents itself as valuable initial groundwork, open to discussion and extensions.

KEYWORDS

Recommender Systems, Reinforcement Learning, Task Environment, Reproducibility, Evaluation.

1 INTRODUCTION

Recommender Systems (RS) help users discover interesting products by means of suggestions. Conventionally, the RS problem has been formalized as a Supervised Learning task (Batch Learning). However, in recent years these foundations have been questioned in light of more complex application settings, such as learning from interactive feedback in fast-paced and dynamic scenarios. Recent works, as explained in [3], have proposed that Reinforcement Learning (RL) can be a more appropriate paradigm (Online and Incremental Learning) to frame the "modern" and Interactive RS problem. Under this paradigm, the RS is viewed as a sequential decision-making Agent focused on learning over time how to perform *actions* (e.g., offer item suggestions), in different *states* (e.g., to different users), using interactive feedback in the form of *reward* signals (e.g., user ratings). Figure 1(a), shows how the RL framework can be used to represent a RS problem (more in [3]).

An important component in RL is the Task Environment, which captures the characteristics of the Agent's application domain and expresses the task/goals the Agent is trying to achieve. Currently, the RS community has mostly focused on creating and sharing datasets and simulations as part of sharing experiments and promoting reproducible research. We argue that this is not enough, and that Environments can encapsulate more assumptions and design decisions that are necessarily incurred in RS evaluation design. In short, when sharing only datasets and simulations many evaluation design decisions are typically left to interpretation, which deeply hinders reproducible research.

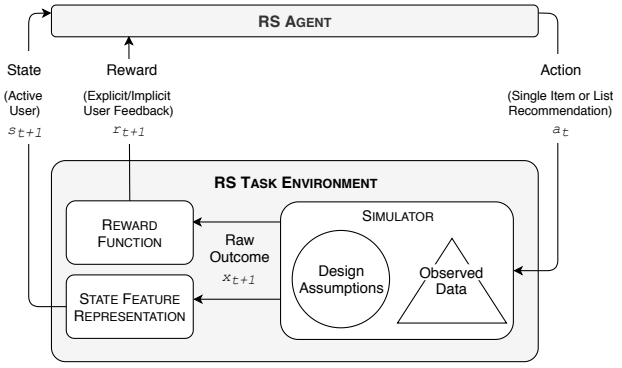

Figure 1: The Interactive Recommender Systems (RS) problem as Reinforcement Learning (RL).
The Figure serves two purposes: (a) Present how the RS problem can be mapped to the RL Framework [23], showing examples of possible states, rewards and actions in the RS field; (b) Focus on the RS Task Environment and present its main components.

2 WHAT IS A TASK ENVIRONMENT?

A *RS Task Environment* (or simply *RS Environment*) is the domain or application scenario where the RS Agent will be deployed, and embodies the most relevant characteristics of the considered problem setting. The Environment is where the Agent operates, and includes everything external to the Agent [23]. In its minimal form, the Environment is a function that for any action a_t performed by the Agent over the current Environment state s_t, will generate a reward r_{t+1} and the next Environment state s_{t+1}, i.e.: $(s_t, a_t) \mapsto (r_{t+1}, s_{t+1})$.

In RL, it is common to describe Environments as a Markov Decision Process (MDP) [23]. This is useful because independent of the specific implementation details: From the Agent's perspective (in our case the RS) the Environment has a simple interface reduced to the information found in states, actions and rewards.

3 RS ENVIRONMENT MAIN COMPONENTS

Although we can describe Environments using MDP constructs, in this work, we propose an alternative general framework of components that can help define a principled way to build Environments

Included in the Offline Evaluation for Recommender Systems Workshop (REVEAL'19), collocated with ACM RecSys 2019.
REVEAL'19, September 20th, 2019, Copenhagen, Denmark.

in the RS field. Figure 1(b) presents the main components a RS Task Environment should define, which are:

Simulator. The *Simulator* imitates the dynamics of the RS application domain (e.g., dynamics of users, items and contextual factors). It takes as input the RS Agent's action a_t (e.g., recommended items) and outputs a *Raw Outcome* x_{t+1} representing all variables observed after the action is performed over the application. In short, it defines the movement from one state of the world to the next given an action. Note that the Simulator only defines world dynamics (e.g., the next active user) and not the Agent's task/goals. To build a Simulator, it is common to use *Observed Data*[1](sampled from the real application) augmented with *Design Assumptions* about the world being modeled. For instance, if the Simulator is based on a static dataset (e.g., a user-item matrix), the simulation designer would have to make design assumptions (ideally, based on prior domain knowledge) about the frequency and order in which users interact with the RS. More examples are presented in the following section.

Reward Function. The *Reward Function* uses the Simulator's output x_{t+1} to generate a bounded numeric reward r_{t+1}. Fundamentally, rewards define the RS goals and are used to motivate the RS Agent to learn an optimal way to act. In fact, typically the RS Agent's goal (objective function) is to maximize a measure of the total reward collected over its interactions with the Environment. Thus, designing the Reward Function is one of the most critical tasks of defining an Environment. In RS, it is common to abstract from x_{t+1} the user's immediate feedback on a recommendation (e.g., explicit/implicit rating) to use as the reward signal. Other measures can also be used, e.g., number of items sold, abandonment rate, conversion rate, session length or revenue among others.

All in all, it can be particularly challenging to translate high-level complex RS goals to an individual numeric reward signal [9, 11], and there is much work to be explored in the field of reward engineering for RS. Possible topics are: designing/modeling goals [8, 15], reward hacking [2, 16, 23], incorporating artificial curiosity/intrinsic motivation [4, 17, 20, 21] (e.g., by explicitly rewarding the RS for learning new information about users) and inverse RL [14].

State Feature Representation. The *State Feature Representation* component takes as input the Simulator's output x_{t+1} to generate state s_{t+1}. A state[2] encapsulates all the relevant information about the Environment made available to the Agent at a given time step to enable decision making. An example state in RS, can be represented with a set of *state variables* that include: active user profile (can be an ID), available items (item IDs and possibly features) and information on contextual factors (e.g., time, location). Though RS feature engineering is a fairly mature topic, it is important to fully share how features are identified, selected and represented. Encapsulating this process in a shareable Environment can help support reproducible evaluation.

[1]Alternative names include empirical data, ground-truth data and logged data.
[2]Also called context in Contextual Bandits, such as [12].

4 WHY SHARE RS ENVIRONMENTS?

RS address a wide variety of complex problem settings. Naturally, there are no perfect simulators or complete datasets that can entirely represent all the characteristics of a specific RS problem setting. As a consequence, evaluation design involves many choices and assumptions, that are not always obvious and are often not made explicit. RS Task Environments can help encapsulate and share many of the complexities and assumptions behind design choices to support reproducible evaluation. In principle, by sharing Environments, we could ensure that solution approaches are in fact being tested and compared for the same RS task specification.

To illustrate choice complexity, we will offer an example of design choices faced when building a Simulator. Let us assume we are building a Simulator based on a ground-truth dataset (Observed Data) with data properties that are closely representative of our use case of interest. When using datasets, we typically face two challenges to build a Simulator:

Data Bias. It is well-known that RS datasets can be biased in multiple ways. A notable example is selection bias [19]; as data about users is ordinarily not collected at random (i.e., data is missing-not-at-random [22]). For instance, users mostly provide feedback on displayed items, which are naturally influenced by the deployed RS (algorithmic bias [1, 6]). Even if items are presented at random, users can choose to provide, or not, item feedback. Note that other context biases can influence users, such as presentation and position bias [7].

Missing Data. Datasets can be missing unobserved confounding variables which can lead to biased results [5, 7]. Moreover, it is natural for users to interact with only a small portion of available items, hence datasets really only represent an incomplete picture of user preferences (such as, only bandit feedback [24]). Mainly, evaluation is difficult (sometimes even unfeasible) if actions taken by the new solution approach do not overlap (at least enough) with the actions previously taken by the RS used for data collection [7].

In this case, the Design Assumptions circle in Figure 1(b) represents the experiment designer's decisions that aim to: correct bias and complete the data. There are multiple paths to achieve these goals depending on the dataset. Let's focus on dealing with missing data. We could [7, 10]: (a) impute missing rating information assuming data is missing at random, or (b) compute counterfactual/off-policy estimators, based on the availability of propensity scores or depending on certain assumptions regarding the data collection strategy [12, 13, 19]. If we move forward with our example, we will soon realize that each decision opens the door to further decisions, each with their own set of implications and conditions for success which impact evaluation results in different (sometimes unknown) ways. Moreover, the bigger the Design Assumptions circle, the more bias is introduced to the RS Task Environment. Nonetheless, evaluation design assumptions are unavoidable.

This is one of many examples of design decision paths that need to be explicitly shared towards building reproducible experiments. We believe building RS Task Environments to define and share RS tasks and problem settings can be a reliable solution, where design assumptions are necessarily made explicit.

5 DISCUSSION AND NEXT STEPS

Describing Interactive RS evaluations in a way which is *complete* and *consistent* to be *efficiently reproduced* and *compared*, can quickly turn into an overwhelming task. In many ways, RS evaluation design is more an art than a science. *RS Task Environments* can be a way to encapsulate and share many of the complexities and design assumptions necessarily made when doing RS evaluations. Recent works such as [3, 18] are steps in this direction. Specifically, this paper introduces a general and modular framework that identifies the main components of a RS Environment, i.e., the shareable components that sufficiently defines a RS problem setting. The paper also motivates the need to build and share RS Environments to support reproducible evaluation.

We have barely scratched the surface of possibilities, and there are multiple paths for *future research*, such as: (a) *A Framework to share RS Environments*: BEARS [3] presents a possible solution, (b) *Defining types of RS Environment:* Individual components can be built and combined to create multiple types of RS Environments. Naturally, a Simulator that is not based on Observed Data (i.e., big Design Assumptions circle in Figure 1(b), aka higher model bias) would have different properties compared to a Simulator that only uses Observed Data (e.g., an online experiment requiring less assumptions about how data is generated, aka lower bias). Also, different Reward Function and State Feature Representation components can be designed for the same Simulator component. Though there seem to be infinite possibilities, we expect that these types of Environments share some common properties and that it will be valuable to create a taxonomy of RS Environments, for instance, to understand which types of Environments are comparable, (c) *Assessing RS Environment Quality*: We cannot build perfect Environments, thus it is important to analyze Environments to understand in which ways they are imperfect and which problem settings they represent best (e.g., to identify benchmark RS Environments for different domains and/or tasks). Note that different types of Environments lead to different bias-variance trade-offs which need to be understood. Overall, we hope that the introduced model helps structure a discussion around the meaning of Task Environments for RS, their role in evaluation design and encourages new research in this area.

ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, cofunded by the European Regional Development Fund.

REFERENCES

[1] Gediminas Adomavicius, Jesse Bockstedt, Shawn Curley, and Jingjing Zhang. 2014. De-biasing user preference ratings in recommender systems. In *RecSys 2014 Workshop on Interfaces and Human Decision Making for Recommender Systems (IntRS 2014)*. 2–9.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[3] Andrea Barraza-Urbina, Georgia Koutrika, Mathieu D'Aquin, and Conor Hayes. 2018. BEARS: Towards an Evaluation Framework for Bandit-based Interactive Recommender Systems. In *Proceedings of the Workshop on Offline Evaluation for Recommender Systems (REVEAL'18). Workshop Programme of the 12th ACM Conference on Recommender Systems (RecSys)*.

[4] Andrew G Barto. 2013. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*. Springer, 17–47.

[5] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research* 14, 1 (2013), 3207–3260.

[6] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. 2017. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. *arXiv preprint arXiv:1710.11214* (2017).

[7] Katja Hofmann, Lihong Li, Filip Radlinski, et al. 2016. Online evaluation for information retrieval. *Foundations and Trends® in Information Retrieval* 10, 1 (2016), 1–117.

[8] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *International Conference on Machine Learning*. 2112–2121.

[9] Dietmar Jannach and Gediminas Adomavicius. 2016. Recommendations with a Purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 7–10. https://doi.org/10.1145/2959100.2959186

[10] Nan Jiang and Lihong Li. 2015. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722* (2015).

[11] Joseph A. Konstan. 2018. From User Experience to Offline Metrics and Back Again. In *Proceedings of the Offline Evaluation for Recommender Systems Workshop (REVEAL'18)*.

[12] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 661–670.

[13] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 297–306.

[14] Ziming Li, Julia Kiseleva, Maarten de Rijke, and Artem Grotov. 2017. Towards learning reward functions from user interactions. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 289–292.

[15] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. 2017. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341* (2017).

[16] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[17] Pierre-Yves Oudeyer and Frederic Kaplan. 2009. What is intrinsic motivation? A typology of computational approaches. *Frontiers in neurorobotics* 1 (2009), 6.

[18] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. 2018. RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising. *arXiv preprint arXiv:1808.00720* (2018).

[19] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. *arXiv preprint arXiv:1602.05352* (2016).

[20] Satinder Singh, Richard L Lewis, and Andrew G Barto. 2009. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*. 2601–2606.

[21] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. 2010. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development* 2, 2 (2010), 70–82.

[22] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 713–722.

[23] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[24] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.