UserSimCRS: A User Simulation Toolkit for Evaluating Conversational Recommender Systems

Jafar Afzali
University of Stavanger
j.afzali@stud.uis.no

Aleksander Mark Drzewiecki
University of Stavanger
am.drzewiecki@stud.uis.no

Krisztian Balog
University of Stavanger
Stavanger, Norway
krisztian.balog@uis.no

Shuo Zhang
Bloomberg
London, United Kingdom
szhang611@bloomberg.net

ABSTRACT

We present an extensible user simulation toolkit to facilitate automatic evaluation of conversational recommender systems. It builds on an established agenda-based approach and extends it with several novel elements, including user satisfaction prediction, persona and context modeling, and conditional natural language generation. We showcase the toolkit with a pre-existing movie recommender system and demonstrate its ability to simulate dialogues that mimic real conversations, while requiring only a handful of manually annotated dialogues as training data.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Conversational recommender systems; user simulation

1 INTRODUCTION

Conversational recommender systems (CRSs) elicit user preferences via multi-turn real-time interactions using natural language [6, 9]. There has been a great deal of progress in recent years on various aspects, including question-based user preference elicitation [5, 10, 29], multi-turn conversational recommendation strategies [12], and natural language understanding and generation [13, 27]. A major challenges that remains, however, is evaluation [6]. Due to the dynamic nature of interactions, measuring performance on

the conversation level is not possible using offline test collections. While online evaluation with users of a live service is an option, it is expensive and does not scale. A promising solution to these issues is *user simulation* [1, 6]. The idea there is to build a simulated user that mimics how a real human would respond in a given dialogue situation [19, 25]. Simulation thus offers a repeatable and reproducible means of evaluation. (We note that it is not meant to replace, but rather to complement human evaluation.)

There is indeed an emerging focus in recent research on using simulation for evaluating conversational information access systems in general [1, 3, 17, 20, 23] and conversational recommenders in particular [25, 26]. The current work aims to contribute to the development of novel CRSs by recognizing the need for better tooling for user simulation. In particular, we provide an extensible open-source toolkit that is designed specifically for evaluation. Our work is unique in at least three regards. First, it focuses on the task of *conversational* recommendation and hence place a strong emphasis on both the recommendation-specific conversation flow and on the human-likeness of the generated user utterances. Second, it centers around *evaluation* as opposed to other uses of simulation (most commonly, synthetic data generation for reinforcement learning). Third, it is designed to work with *existing CRSs*, without needing access to source code or knowledge of their inner workings. It merely requires collecting and annotating a small sample of dialogues.

Building on an established agenda-based simulator [25], we introduce novel components, motivated by recent research [17, 23, 26], for modeling user satisfaction, persona and context, and conditional natural language generation. Given its modular design, the toolkit can also be easily extended with other modeling options or additional components. The toolkit is comprised of two Python libraries, which are made publicly available on GitHub: DialogueKit[1] is a collection of generic and reusable dialogue components, and UserSimCRS[2] is an extensible user simulator built on top.

2 RELATED WORK

While there are several efforts on simulation toolkits for recommender systems [8, 11, 14, 16, 21], our work differs from those in two major ways. First, we focus on the task of *conversational* recommendations and hence place a strong emphasis on natural language understanding and generation. Thus, unlike others that operate in

---

[1] https://github.com/iai-group/DialogueKit
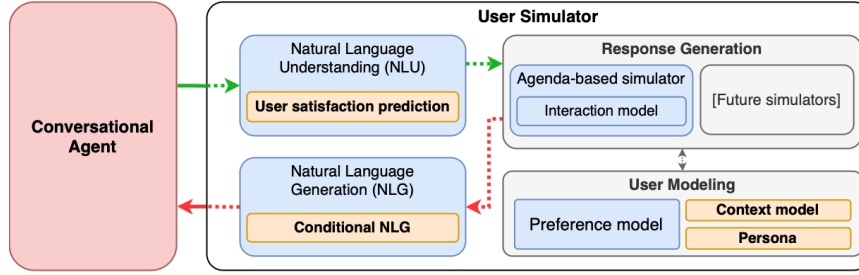[2] https://github.com/iai-group/UserSimCRS

Figure 1: Conceptual overview of the user simulator. The parts in blue follow [25], while the yellow ones are novel additions.

the "intent space," we operate in the "language space." Second, our objective is *system evaluation*, as opposed to training end-to-end recommender systems using reinforcement learning (RL).

Our toolkit implements an agenda-based simulator [18], building on and extending the approach in [25]. Alternatively, model-based simulation could also be employed as it has been done recently for task-based dialogue systems. Shi et al. [22] demonstrate how to build model-based user simulators that rely on a simple Seq2seq dialogue system with copy and attention mechanisms, to facilitate RL-based dialogue system training. ConvLab-2 [28] is an open-source toolkit that enables researchers to build task-oriented dialogue systems, where user simulators are provided to support end-to-end evaluation. These simulators can be assembled by equipping a neural network-based user policy with NLU and NLG components. Tseng et al. [24] propose a learning framework for developing dialogue systems that perform joint optimization with an LSTM-based user simulator, which consists of a dialogue manager, an NLG model, and a dialogue context encoder. The dialogue systems and user simulator models are pre-trained using supervised learning and then fine-tuned using reinforcement learning based on the generated dialogues. Importantly, such model-based approaches can also be incorporated into our framework in the future.

3 CONCEPTUAL OVERVIEW

The goal of user simulation is to mimic how real users would respond in given dialogue situation [19, 25]. Conceptually, our user simulator follows the architecture of a typical task-based dialogue system, which consists of natural language understanding, response generation, and natural language generation components. Additionally, there is a dedicated user modeling component; see Fig. 1. We opt for a modular design, as opposed to an end-to-end trainable system, in order to have complete control over how responses are generated and to allow for flexible extensions. Our work builds on and extends the approach proposed in [25] as detailed below.

Natural language understanding (NLU) is responsible for obtaining a structured representation of text utterances. Conventionally, it entails intent classification and entity recognition. Additionally, motivated by recent research [17, 23], we also include a classifier for user satisfaction prediction.[3]

Response generation is currently based on agenda-based simulation [18], however, it could be replaced with other approaches in the future. Following [25], response generation is based on an *interaction model*, which is responsible for initializing the agenda and

updating it. Updates to the agenda can be summarized as follows: if the agent responds in an expected manner, the interaction model *pulls* the next action off the agenda; otherwise, it either repeats the same action as the previous turn or samples a new action.

User modeling consists of three sub-components. The *preference model* captures users' likes and dislikes. Following [25], it is modeled as a *personal knowledge graph* [2], where nodes correspond to items and attributes. Novel to our work is the modeling of *persona*, which can capture user-specific traits, e.g., patience or cooperativeness, and *context*, which can characterize the situation of the user, e.g., temporal (time of the day and weekday vs. weekend), relational (alone vs. group setting), or conversational (user satisfaction). We focus on contextual aspects as these represent a so far unexplored area of user modeling [9] and there is evidence suggesting that language usage depends on persona and context [15, 23].

Natural language generation (NLG) is currently template-based, that is, given the output of the response generation module, a fitting textual response is chosen and may be instantiated with preferences. Additionally, we extend the NLG such that it can be conditioned on context. For example, user responses might be shorter/longer depending on the time of the day or users could use a stronger language when getting dissatisfied with the system.

4 SOFTWARE ARCHITECTURE

The toolkit is written in Python and is based on a modular architecture to support additional components, different models, and custom features to be added in the future. There are two main libraries that are stacked on each other: **DialogueKit** provides basic dialogue management functionalities, while **UserSimCRS** contains simulation-specific models and logic. See Fig. 2 for an overview of the main packages and their dependencies. Both libraries are made available in the Python Package Index (PyPI).

4.1 DialogueKit

DialogueKit models *dialogue participants* (users and agents), *domains* (which define the types of slots for a particular application), *utterances*, and *annotations* as base concepts. Utterances may be annotated with *intents* and *slot-value* pairs. DialogueKit currently supports two models for annotation, a cosine classifier for intents and a minimal pipeline DIET classifier [4] for slot-value pairs.[4] A *dialogue connector* is included to orchestrate and store the conversation between participants (human-human, human-machine, or

---

[3]User satisfaction prediction is only used in the training stage to annotate dialogues.

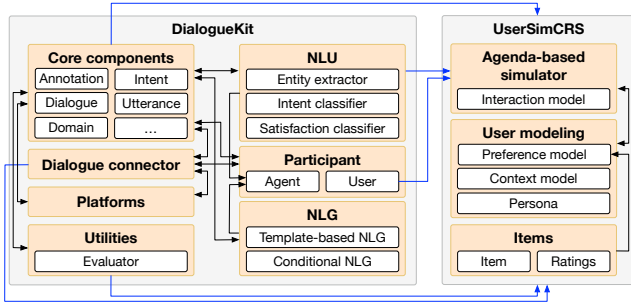[4]The DIET classifier can be used for intent detection as well.

Figure 2: Overview of the main packages (in yellow) with some of the core modules highlighted (in white). Arrows indicate intra-library dependencies (in blue) and inter-library dependencies (in black).

machine-machine). Furthermore, the *evaluation* component provides functionality required to evaluate a set of conversations with respect to standard metrics (such as AvgTurns and AvgSuccess).

4.2 UserSimCRS

The UserSimCRS library implements the simulation-specific components in Fig. 1, specifically, response generation and user modeling. During a conversation, any time the user is asked to provide preferences, the preference model is consulted. Context is modeled in a generic way such that it can capture, among others, temporal, relational, and conversational factors. The generation of user utterances may be conditioned on the user's context and persona. Next, we elaborate on how to use UserSimCRS for system evaluation. Note that the library may also be used for training agents, but that is outside the focus of the current paper.

5 SYSTEM EVALUATION USING SIMULATION

This section discusses how to employ simulation for evaluating an existing CRS and illustrates this with a case study.

5.1 Methodology

The main objective of simulation-based evaluation in this work is to establish a *relative comparison* between two systems. These may be different variants of the same CRS or two different systems. Importantly, the user simulator needs to target the differences that we care about. For the sake of illustration, assume that there is a baseline conversational movie recommender that understands movie genres and an improved version that also recognizes plot keywords. Having a user simulator that asks only for genres but not for plot keywords will not capture the differences between these two systems. Therefore, as a general principle, the user simulator needs to be co-developed with the CRS and customized to mimic the targeted user behavior.

5.2 Setting up Simulation

A unique feature of our toolkit is that it allows for the evaluation of any existing CRS by treating it as a "black box." That is, it does not require access to the source code or assume knowledge of its inner workings—it merely relies on observable behavior. Setting up an existing CRS with our simulator involves the following steps:

(1) **Prepare domain and item collection**: A config file with domain-specific slot names must be prepared for the preference model. Additionally, a file containing the item collection is required.

(2) **Provide preference data**: Preference data is consumed in the form of item ratings (user ID, item ID, and rating triples).

(3) **Dialogue sample**: A small sample of dialogues with the CRS needs to be collected. The sample size depends on the complexity of the system, in terms of action space and language variety, but is generally in the order of 5-50 dialogues.

(4) **Define interaction model**: A config file containing the space of user and agent intents (i.e., possible actions), as well as the set of expected agent responses for each user intent, is required for the interaction model. The baseline (CRSv1) interaction model shipped with the UserSimCRS library offers a starting point, which may be further tailored according to the behavior and capabilities of the given CRS.

(5) **Annotate sample**: The sample of dialogues must contain utterance-level annotations in terms of intents and entities, as this is required to train the NLU and NLG components. Note that the slots used for annotation should be the same as the ones defined in the domain file (cf. Step 1) and intents should follow the ones defined in the interaction model (cf. Step 4.).

(6) **Define user model/population**: Simulation is seeded with a user population that needs to be characterized, for example, in terms of the different contexts (e.g., weekday vs. weekend, alone vs. group setting) and personas (e.g., patient and impatient users). Further, the number of users to be generated is to be specified. Each user will have their own preference model, which may be instantiated by grounding it in actual preferences (i.e., the ratings dataset given in Step 2).

(7) **Train simulator**: The NLU, NLG, and response generation components of the simulator are trained using the annotated dialogue sample.

(8) **Run simulation**: Running the simulator will generate a set of simulated conversations for each user with the CRS and save those to files.

(9) **Perform evaluation**: Evaluation takes the set of simulated dialogues generated in the previous step as input, and measures the performance of the CRS in terms of the metrics implemented in DialogueKit.

5.3 Case Study

To see our user simulator in action, we conducted a case study with IAI MovieBot [7],[5] which is an open-source conversational movie recommender system. This required creating a connector agent in DialogueKit, which can talk to IAI MovieBot via a RESTful API. We followed the steps listed above to prepare the user simulator. This included collecting a sample of 8 dialogues, configuring the domain (with *title*, *genre*, and *keyword* as slots), and annotating user and system utterances using intents (according to our CRSv1 interaction model) and slot-value pairs. As it can be seen from the sample dialogue in Fig. 3, the simulator could successfully complete dialogues with the CRS, mimicking the behavior of users observed in the training data it was exposed to.

---

[5]https://github.com/iai-group/MovieBot

Hi there. I am IAI MovieBot, your movie recommending buddy. I can recommend you movies based on your preferences. I will ask you a few questions and based on your answers, I will try to find a movie for you.",

DISCLOSE.NON-DISCLOSE

Hello.

DISCLOSE.NON-DISCLOSE

Do you have any specific genres in mind?

INQUIRE.ELICIT

How about Crime?

DISCLOSE

There are almost 5300 crime films. Please answer a few more questions to help me find a good movie.Can you give me a few keywords? For example, 'organized crime' or 'contact lens'.",

INQUIRE.ELICIT

Pixar animation

DISCLOSE

Do you like: Cat City 8.4 96 min

REAVEAL

Tell me something about it

INQUIRE.ITEMINFO

What would you like to know about Cat City?

INQUIRE.MORE

Director name

INQUIRE.MORE

Its directed by Béla Ternovszky.

DISCLOSE.MORE

I like this recommendation

NOTE.ACCEPT

Please choose your next step.

INQUIRE.NEXT

I would like to quite now.

COMPLETE

I hope you had a good experience. Bye.

END

Figure 3: Sample dialogue between IAI MovieBot (Left, in green) and the user simulator (Right, in blue).

6 CONCLUSION AND FUTURE DIRECTIONS

We have presented a user simulation toolkit, organized into two Python libraries around general dialogue management and specific user simulation functionality, to facilitate research on both conversational recommender systems and simulation-based evaluation. The toolkit is shipped with solid baseline models for each of the components, a detailed set of instructions, and a working example with an existing CRS, in order to make it easy for researchers and developers to start conducting simulation-based experiments. Future work is concerned with extending the components with additional modeling options, including alternatives to agenda-based simulation. We also plan to evaluate additional existing CRSs to ensure that our framework generalizes to diverse systems.

REFERENCES

[1] Krisztian Balog. 2021. Conversational AI from an Information Retrieval Perspective: Remaining Challenges and a Case for User Simulation. In *Proc. of DESIRES '21.* 80–90.

[2] Krisztian Balog and Tom Kenter. 2019. Personal Knowledge Graphs: A Research Agenda. In *Proc. of ICTIR '19.* 217–220.

[3] Krisztian Balog, David Maxwell, Paul Thomas, and Shuo Zhang. 2021. Report on the 1st Simulation for Information Retrieval Workshop (Sim4IR 2021) at SIGIR 2021. *SIGIR Forum* 55, 2, Article 10 (dec 2021).

[4] Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. 2020. DIET: Lightweight Language Understanding for Dialogue Systems. arXiv:2004.09936 [cs.CL]

[5] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. In *Proc. of KDD '16.* 815–824.

[6] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *AI Open* 2 (2021), 100–126.

[7] Javeria Habib, Shuo Zhang, and Krisztian Balog. 2020. IAI MovieBot: A Conversational Movie Recommender System. In *Proc. of CIKM '20.* 3405–3408.

[8] Eugene Ie, Chih wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. RecSim: A Configurable Simulation Platform for Recommender Systems. arXiv:1909.04847 [cs.LG]

[9] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A Survey on Conversational Recommender Systems. *ACM Comput. Surv.* 54, 5 (2021).

[10] Ivica Kostric, Krisztian Balog, and Filip Radlinski. 2021. Soliciting User Preferences in Conversational Recommender Systems via Usage-Related Questions. In *Proc. of RecSys '21.* 724–729.

[11] Karl Krauth, Sarah Dean, Alex Zhao, Wenshuo Guo, Mihaela Curmei, Benjamin Recht, and Michael I. Jordan. 2020. Do Offline Metrics Predict Online Performance in Recommender Systems? arXiv:2011.07931 [cs.LG]

[12] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *Proc. of WSDM '20.* 304–312.

[13] Raymond Li, Samira Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Proc. of NIPS '18.* 9748–9758.

[14] Martin Mladenov, Chih-Wei Hsu, Vihan Jain, Eugene Ie, Christopher Colby, Nicolas Mayoraz, Hubert Pham, Dustin Tran, Ivan Vendrov, and Craig Boutilier. 2021. RecSim NG: Toward Principled Uncertainty Modeling for Recommender Ecosystems. arXiv:2103.08057 [cs.LG]

[15] Namkee Park, Kyungeun Jang, Seonggyeol Cho, and Jinyoung Choi. 2021. Use of Offensive Language in Human-Artificial Intelligence Chatbot Interaction: The Effects of Ethical Ideology, Social Competence, and Perceived Humanlikeness. *Comput. Hum. Behav.* 121 (2021), 106795.

[16] 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:1808.00720 [cs.IR]

[17] Alexandre Salle, Shervin Malmasi, Oleg Rokhlenko, and Eugene Agichtein. 2021. Studying the Effectiveness of Conversational Search Refinement Through User Simulation. In *Proc. of ECIR '21.* 587–602.

[18] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proc. of NAACL '07.* 149–152.

[19] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *Knowl. Eng. Rev.* 21, 2 (June 2006), 97–126.

[20] Ivan Sekulić, Mohammad Aliannejadi, and Fabio Crestani. 2022. Evaluating Mixed-Initiative Conversational Search Systems via User Simulation. In *Proc. of WSDM '22.* 888–896.

[21] Bichen Shi, Makbule Gulcin Ozsoy, Neil Hurley, Barry Smyth, Elias Z. Tragos, James Geraci, and Aonghus Lawlor. 2019. PyRecGym: A Reinforcement Learning Gym for Recommender Systems. In *Proc. of RecSys '19.* 491–495.

[22] Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. 2019. How to Build User Simulators to Train RL-based Dialog Systems. In *Proc. of EMNLP-IJCNLP '19.* 1990–2000.

[23] Weiwei Sun, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. 2021. Simulating User Satisfaction for the Evaluation of Task-Oriented Dialogue Systems. In *Proc. of SIGIR '21.* 2499–2506.

[24] Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable Dialogue Systems and User Simulators. In *Proc. of ACL '21.* 152–166.

[25] Shuo Zhang and Krisztian Balog. 2020. Evaluating Conversational Recommender Systems via User Simulation. In *Proc. of KDD '20.* 1512–1520.

[26] Shuo Zhang, Mu-Chun Wang, and Krisztian Balog. 2022. Analyzing and Simulating User Utterance Reformulation in Conversational Recommender Systems. In *Proc. of SIGIR '22.* 133–143.

[27] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. In *Proc. of CIKM '18.* 177–186.

[28] Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *Proc. of ACL '20.* 142–149.

[29] Jie Zou, Yifan Chen, and Evangelos Kanoulas. 2020. Towards Question-Based Recommender Systems. In *Proc. of SIGIR '20.* 881–890.