RecWizard: A Toolkit for Conversational Recommendation with Modular, Portable Models and Interactive User Interface

Zeyuan Zhang*, Tanmay Laud*, Zihang He*, Xiaojie Chen, Xinshuang Liu, Zhouhang Xie, Julian McAuley, Zhankui He

University of California, San Diego
{zez018, tlaud, z6he, xic051, xil235, zhx022, jmcauley, zhh004}@ucsd.edu

Abstract

We present a new Python toolkit called **`RecWizard`** for Conversational Recommender Systems (CRS). **`RecWizard`** offers support for development of models and interactive user interface, drawing from the best practices of the Huggingface ecosystems. CRS with **`RecWizard`** are *modular, portable, interactive* and *Large Language Models (LLMs)-friendly*, to streamline the learning process and reduce the additional effort for CRS research. For more comprehensive information about **`RecWizard`**, please check our GitHub https://github.com/McAuley-Lab/RecWizard.

Introduction

Conversational Recommender Systems (CRS) (Christakopoulou, Radlinski, and Hofmann 2016; Li et al. 2018) are gaining increasing attention from industry and academia, especially with the emergence of Large Language Models (LLMs) (Friedman et al. 2023; He et al. 2023; Wang et al. 2023). To expedite CRS research, there is a pressing need for an open-source toolkit that lowers the barrier to reusing CRS and LLMs resources, developing new CRS, and interacting via user interface for debugging or evaluation. However, current toolkits fail to meet such requirements (Miller et al. 2017; Zhou et al. 2021; Quan et al. 2022).

To address these limitations, we propose **`RecWizard`**, a Hugging Face (Wolf et al. 2020) (HF)-based CRS toolkit for research purposes, with the following properties:

- **Modular**: We abstract CRS to a lower *module level* and a higher *pipeline level* and make them modular.
- **Portable**: With HF compatibility, modules and pipelines can be effortlessly shared online and easily deployed.
- **Interactive**: A user-friendly interactive interface is provided for conversations between users and CRS.
- **LLMs-Friendly**: We demonstrate LLMs as different roles, e.g., recommender modules, in RecWizard.

Related Works

Traditional recommender-system toolkits (Zhao et al. 2021, 2022; Ivchenko et al. 2022) only support users' actions (e.g., clicks or purchases), not CRS, and general-purpose conversational-system toolkits (Miller et al. 2017) are not designed for CRS use cases. We outline the existing CRS toolkits (Zhou et al. 2021; Quan et al. 2022) and highlight the new properties in our RecWizard in Table 1.

	CRSLab	FORCE	Ours.
Open-Source?	✓	✗	✓
Modular?	*?*	✓	✓
Portable?	✗	✗	✓
User Interface?	✗	✓	✓
LLM-Supported?	✗	✗	✓
CRS Setting	Flexible	Rule-based	Flexible

Table 1: The properties of **`RecWizard`** (ours.) compared to two previous CRS related toolkits, CRSLab (Zhou et al. 2021) and FORCE (Quan et al. 2022)

Design Principles

We design an abstraction for CRS with RecWizard at two levels, as shown in Figure 1 (a):

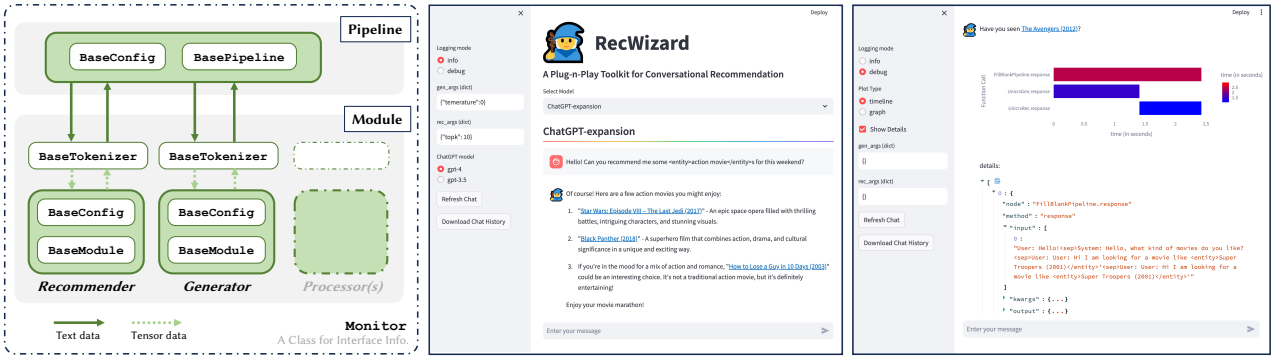
1. **Module level:** We offer a *recommender* module for recommendations and a *generator* module for natural-language responses, as many papers implemented (Li et al. 2018; Chen et al. 2019; Wang et al. 2022). Further, *processor* modules can be integrated to extract vital information, e.g., via *entity linking*, from users' raw text.
2. **Pipeline level:** A RecWizard pipeline is a high-level logic that determines when and how to call the modules and how to aggregate the results from those modules.

By default, all modules communicate using natural language (**text data**), ensuring maximum modularity. However, we also provide developers with the lower-level methods to define module communication through **tensor data** in a flexible and differentiable manner.

Why Use RecWizard

In this section, we present RecWizard use-cases for flexible *development* and deploying *interactive user interface*.

*These authors contributed equally.



(a) RecWizard Architecture (b) User Interface: INFO Mode (c) User Interface: DEBUG Mode

Figure 1: **(a)** The RecWizard architecture comprises pipeline and module levels. Text data flows between modules and the pipeline, while tensor data flows within modules after being processed by RecWizard tokenizers, ensuring RecWizard modularity and portability. **(b)** In INFO mode, the RecWizard example *ChatGPT-expansion* includes the ReDIAL-Rec recommender module and ChatGPT with prompts as the generator module. Users can select models, set basic arguments, and chat with RecWizard. **(c)** In DEBUG mode, using the *UniCRS-ReDIAL* model as an example, module-level timeline visualization and intermediate messages are enabled for debugging or in-depth demonstrations in the user interface.

Effortless Usage

Module Level: CRS practitioners can easily load any trained recommender (or generator) modules from the HF hub. For example, a UniCRS (Wang et al. 2022) recommender module pretrained on ReDIAL dataset can be loaded seamlessly (Li et al. 2018):

```
unicrs_rec = recwizard.UnicrsRec \
          .from_pretrained('unicrs-rec-redial')
```

Pipeline Level: To build a RecWizard model, the next step is loading a high-level pipeline. Apart from loading a pipeline with a one-liner similar to the module-level example above, it is also flexible to create UniCRS model variants by combining different modules under a certain pipeline. Here we use a new ChatGPT (Schulman et al. 2022)-based generator module to build a **`unicrs_variant`**:

```
unicrs_variant = recwizard.FillblankPipeline(
    config=config,
    rec_module=unicrs_rec,
    gen_module=recwizard.ChatGPTGen \
      .from_pretrained('chatgpt-gen-fillblank'))
```

This example highlights the modular, portable, and LLM-friendly design of RecWizard, which simplifies the usage of the trained modules or pipelines.

Flexible Development

RecWizard Tokenizers are the key to our framework, bridging the text interface between modules. At its core, it uses the "composite pattern" to extend HF tokenizers for CRS. Further, it provides a framework to parse information (like entities) in text-only format that we designed for conversational recommendation.

New modules can be specified by defining the configuration, tokenizer, and module classes as below[1]:

```
class NewModule(recwizard.BaseModule):
      # for recommender, generator or more processors
    def __init__(self, ...):
      # build it similar to HF PreTrainedModel
    def forward(self, tensor_inputs, labels):
      # define the flow of tensors through the module
    def response(self, raw_inputs, tokenizer):
      # define inputs and outputs based on forward
```

Pipelines are specified by configuration and **`response`** method that defines the execution flow of the modules. All our Base classes are designed to be compatible with HF's **`push_to_hub`** method.

User-Friendly Interface

We provide an easy-to-use user interface for our models to support two different control modes in Figure 1 (b)-(c):

INFO Mode: Users can chat with a selected system, asking for natural language responses or recommended items. We suggest using this mode for: (1) demonstrating the final RecWizard model; (2) inviting users for human evaluation.

DEBUG Mode: Users observe the module execution timeline, intermediate results, and control the internal module arguments. We suggest this mode for (1) debugging the pipeline at the module level and (2) understanding or explaining how a certain RecWizard pipeline works.

Conclusion and Future Work

We present **`RecWizard`**, a toolkit for Conversational Recommender Systems (CRS) research based on Hugging Face. RecWizard offers user-friendly deployment and development APIs and user interface for conversation interaction and debugging. We plan to share more pipelines and trained modules for CRS research and contribute to CRS benchmarking as well as online services in the future.

[1]We omit the configuration and tokenizer classes due to limited spacing. Please see the complete code template in supplementary.

References

Chen, Q.; Lin, J.; Zhang, Y.; Ding, M.; Cen, Y.; Yang, H.; and Tang, J. 2019. Towards Knowledge-Based Recommender Dialog System. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1803–1813.

Christakopoulou, K.; Radlinski, F.; and Hofmann, K. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 815–824.

Friedman, L.; Ahuja, S.; Allen, D.; Tan, T.; Sidahmed, H.; Long, C.; Xie, J.; Schubiner, G.; Patel, A.; Lara, H.; et al. 2023. Leveraging Large Language Models in Conversational Recommender Systems. *arXiv preprint arXiv:2305.07961*.

He, Z.; Xie, Z.; Jha, R.; Steck, H.; Liang, D.; Feng, Y.; Majumder, B. P.; Kallus, N.; and McAuley, J. 2023. Large language models as zero-shot conversational recommenders. *arXiv preprint arXiv:2308.10053*.

Ivchenko, D.; Van Der Staay, D.; Taylor, C.; Liu, X.; Feng, W.; Kindi, R.; Sudarshan, A.; and Sefati, S. 2022. Torchrec: a pytorch domain library for recommendation systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*, 482–483.

Li, R.; Ebrahimi Kahou, S.; Schulz, H.; Michalski, V.; Charlin, L.; and Pal, C. 2018. Towards deep conversational recommendations. *Advances in neural information processing systems*, 31.

Miller, A. H.; Feng, W.; Fisch, A.; Lu, J.; Batra, D.; Bordes, A.; Parikh, D.; and Weston, J. 2017. ParlAI: A Dialog Research Software Platform. *arXiv preprint arXiv:1705.06476*.

Quan, J.; Wei, Z.; Gan, Q.; Yao, J.; Lu, J.; Dong, Y.; Liu, Y.; Zeng, Y.; Zhang, C.; Li, Y.; et al. 2022. FORCE: A Framework of Rule-Based Conversational Recommender System. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 13215–13217.

Schulman, J.; Zoph, B.; Kim, C.; Hilton, J.; Menick, J.; Weng, J.; Uribe, J. F. C.; Fedus, L.; Metz, L.; Pokorny, M.; Lopes, R. G.; and Zhao, S. 2022. Chatgpt: Optimizing language models for dialogue. *OpenAI*.

Wang, X.; Tang, X.; Zhao, W. X.; Wang, J.; and Wen, J.-R. 2023. Rethinking the Evaluation for Conversational Recommendation in the Era of Large Language Models. *arXiv preprint arXiv:2305.13112*.

Wang, X.; Zhou, K.; Wen, J.-R.; and Zhao, W. X. 2022. Towards unified conversational recommender systems via knowledge-enhanced prompt learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1929–1937.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 38–45.

Zhao, W. X.; Hou, Y.; Pan, X.; Yang, C.; Zhang, Z.; Lin, Z.; Zhang, J.; Bian, S.; Tang, J.; Sun, W.; et al. 2022. RecBole 2.0: towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 4722–4726.

Zhao, W. X.; Mu, S.; Hou, Y.; Lin, Z.; Chen, Y.; Pan, X.; Li, K.; Lu, Y.; Wang, H.; Tian, C.; et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *proceedings of the 30th acm international conference on information & knowledge management*, 4653–4664.

Zhou, K.; Wang, X.; Zhou, Y.; Shang, C.; Cheng, Y.; Zhao, W. X.; Li, Y.; and Wen, J.-R. 2021. CRSLab: An open-source toolkit for building conversational recommender system. *arXiv preprint arXiv:2101.00939*.