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.

<table>
<thead>
<tr>
<th>CRS Setting</th>
<th>CRSLab</th>
<th>FORCE</th>
<th>Ours.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-Source?</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Modular?</td>
<td>?</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Portable?</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>User Interface?</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>LLM-Supported?</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

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.
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):

```python
class UnicrsRec(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.

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 below1:

```python
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
```

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


