Towards an Open, Collaborative REST API for Recommender Systems

Iván García, Alejandro Bellogín
alejandro.bellogin@uam.es

RecSys 2018 – Vancouver, Canada

Main Idea

In this work, we propose and show an example implementation for a common REST API focused on Recommender Systems.

This API meets the most typical requirements faced by Recommender Systems practitioners (adding users, items, and events, providing recommendations) while, at the same time, is open and flexible to be extended, based on the feedback from the community.

We also present a Web client that demonstrates the functionalities of the proposed API.

Future Work

- Discuss (with the community) other features to be added or any interesting modifications
- Integrate more recommendation libraries
- Include the possibility to evaluate the system
- Other clients: mobile apps, other language wrappers
- And more

Contribute! **github.com/abellogin/REST4RecSys/issues**

API Endpoints

URL	Method	Description
user/add	POST	Add a user
user/get/{uid}	GET	Returns a user
user/get	GET	Returns all users
user/delete/{uid}	DEL	Removes a user
user/get/{uid}/events	GET	Returns the events of a user
item/add	POST	Add an item
item/get/{iid}	GET	Returns an item
item/get	GET	Returns all items
item/delete/{iid}	DEL	Removes an item
item/get/{iid}/events	GET	Returns the events of an item
event/add	POST	Add an event
event/get/user/{uid}/item/{iid}	GET	Returns an event
user/get/{uid}/recommendations	GET	Returns recommendations of a user
train	GET	Train the recommender
statistics/get	GET	Returns statistics about the system

Table 1: Selection of most representative API endpoints.

Framework

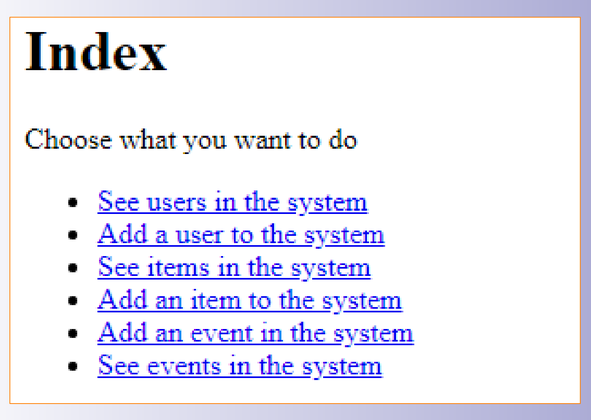
- Backend stores the model (users, items, events) in a very generic way
- Open source libraries based on Java were used to develop the backend (Dropwizard for the Web services and RankSys for the recommendation model)
- API endpoints: based on the proposal in https://web.archive.org/web/20160324042313/http://www.recsyswiki.com:80/wiki/Common_Recommender_REST_API we analysed the typical tasks in a recommender system and defined many URLs, following standard REST principles and patterns.
- Web client: a Web page where most of the API functionalities can be tested

Code examples

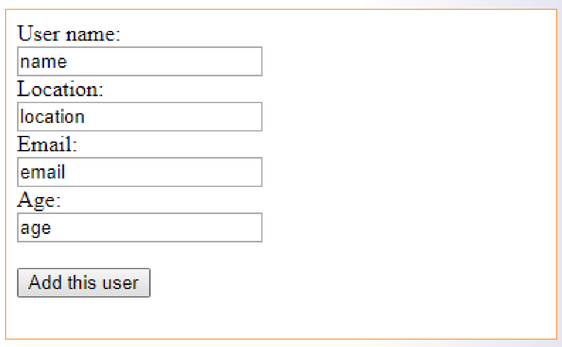
```
@GET
@Path("/item/get/{iid}/events")
@Produces(MediaType.APPLICATION_JSON)
public Collection<Event> getItemEvents(@PathParam("iid") String itemId) {
    return model.getItemEvents(itemId);
}

@POST
@Path("/event/add")
@Consumes(MediaType.APPLICATION_JSON)
public String addEvent(Event e) {
    return addEvent(e.getUid(), e.getIid(), e);
}

@POST
@Path("/event/add/user/{uid}/item/{iid}")
@Consumes(MediaType.APPLICATION_JSON)
public String addEvent(@PathParam("uid") String userId, @PathParam("iid") String itemId,
        Event e) {
    String id = model.addEvent(userId, itemId, e);
    if (model.getAllEvents().size() % DEF_TRAIN_EVENTS == 0) {
        train(null);
    }
    return id;
}

@GET
@Path("/event/get/user/{uid}/item/{iid}")
@Produces(MediaType.APPLICATION_JSON)
public Collection<Event> getEvents(@PathParam("uid") String userId, @PathParam("iid") String itemId) {
    return model.getEvents(userId, itemId);
}
```

Web Client

Index http://localhost:8080/interface/index



Adding a user http://localhost:8080/interface/add/user



Show all events http://localhost:8080/interface/get/events

https://github.com/abellogin/REST4RecSys

Powered by

Dropwizard

RankSys

Acknowledgements: this work was supported by the project TIN2016-80630-P (MINECO).