Рекомендательная система для библиотек. Команда LZRK.

Актуальная версия

- WEB UI: http://159.65.200.185:8000/
- REST API http://159.65.200.185:8000/recommendations//
- API docs: http://159.65.200.185:8000/docs

Стек:

ML:

- pandas
- numpy
- lightfm

Web:

- fastapi

Запуск проекта:

1. Скачать и разархивировать в папку `docker_vol/raw/` архив `raw.zip` отсюда http://159.65.200.185:9000/ Структура `docker_vol/raw/` должна быть следующей:

```
 $ls -c1 raw/
books.json
books_full_df.csv.zip
circulaton_1.csv
circulaton_10.csv
circulaton_11.csv
circulaton_12.csv
circulaton_13.csv
circulaton_14.csv
circulaton_15.csv
circulaton_16.csv
circulaton_2.csv
circulaton_3.csv
circulaton_4.csv
circulaton_5.csv
circulaton_6.csv
circulaton_7.csv
circulaton_8.csv
circulaton_9.csv
dataset_knigi_1.csv
```

`books_full_df.csv.zip` - пересохраненный в CSV и сжатый файл `books_full.jsn` 2. `make build-app` - создание Docker-образа 3. `make prepare-data` - подготовка данных. Рузельтат появится в `docker_vol/prepared/` 4. `make train-lfm` - обучение модели `LightFM`. Модель появится в `docker_vol/models/` 5. `make run-server` - запуск web-сервера. 6. web-интерфейс будет доступен по адресу `http://127.0.0.1:8000/`, REST API `http://127.0.0.1:8000/recommendations/`

Подход к решению:

Был выбран гибридный подход к формированию рекомендаций – основанный на алгоритмическом подходе и коллаборативной фильтрации

Алгоритм формирования рекомендаций:

Коллаборативная фильтрация для первых трех книг:

1. Обучается модель `LightFM` без каких-либо признаков пользователей и книг – классическая коллаборативная фильтрация основанная на факторизации матриц
2. Модель `LightFM` явно не используется. Используется только ее внутренее представление книг для поиска наиболее похожих на нее кандидатов через косинусное расстояние
3. Результаты алгоритмически дофильтровываются

Алгоримический подход для оставшихся двух книг:

1. Выбирается последняя книга, с которой взаимодействовал пользователь
2. Для последних книг из истории итеративно ищутся похожие книги, полученные через коллаборативную фильтрацию – отобранные кандидаты формируют первые 3 книги для рекомендаций. Результаты алгоритмически дофильтровываются.
3. По автору последней книги выбираются самые популярные книги этого автора (учитывается `outputCount`) и предлагаются пользователю – так получаются последние 2 книги для рекомендаций

Новый пользователь и "холодный старт"

Так как о новом пользователе ничего не известно, было принято решение рекомендовать выборку популярных книг из широких рубрик: - 479;Художественная литература - 496;Историческая и приключенческая литература - 534;Литература для детей и юношества - 551;Зарубежная художественная литература для детей и юношества - 511;Фэнтэзи

Структура проекта:

`app.prepare_data` – подготовка данных. Результатом являются файлы: - `circulation_df.pickle` - объединенные и обработанные `circulaton_*.csv` + присоединенные данные из `dataset_knigi_1.csv` - `books_full_df.pickle` - обработанный `books_full_df.csv.zip`. Оставлены книги, которые участвовали во взаимодействиях с пользователями - `interaction_df.pickls` - данные о взаимодействиях пользователей и книг - `knigi_df.pickle` - обработанный файл `dataset_knigi_1.csv`

`app.train_lfm` - скрипт, обучающий модель LightFM. Результатом является файл модели `lfm-model.pickle`

`app.predict.Predictor` – класс отвечающий за предоставление рекомендаций. Он вызывается из `app.server.get_recommendations` при получении API-запроса.

REST API реализован на фреймворке FastAPI, который обладает легковесностью, высоким быстродействием и типизированными интерфейсами