EasyStudy framework

Framework for building user studies

Motivation

- Do not rely solely on offline evaluation
  - Online evaluation is expensive
  - User studies are possible, but it is time consuming to write it from scratch (e.g. showing recommended items, preference elicitation, interaction logging, etc.) We ideally want to make some of those steps reusable.

    ⇒ EasyStudy framework

Technical Details

- Flask application written in Python, sqlite DB
- Modularity achieved using Flask plugins
  - Individual plugins correspond to "templates“ that can be used to create user studies
  - We have an “interface” that has to be satisfied by the plugins (see README in GitHub)
    - Plugin defines some endpoints and the framework ensures their invocation. The plugin then determines their implementation (can be almost anything, e.g. reuse some functionality from other plugin, main app, etc.)

Fastcompare plugin

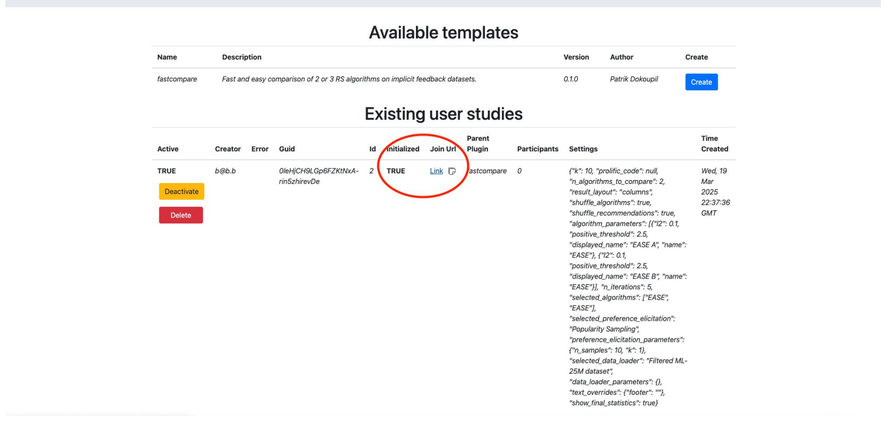
- Fast comparison of RS algorithms
  - Flow: user details ⇒ **pref. Elicitation** ⇒ n iterations of **algorithm comparison** ⇒ final page with statistics.
  - User study can be parameterized:
    - 2 datasets, various RS algorithms, preference elicitation methods, number of iterations, size of recommendation, etc.
    - The plug-in itself is extensible
      - Adding new algorithms, datasets, preference-elicitation methods by subclassing
  - Implicit feedback

Setup

- Repository: https://github.com/pdokoupil/EasyStudy/tree/ndbi021 (**contact email** is there as well)
  - `git clone`, `cd` to the repository, then `checkout` **ndbi021** branch
    - `$env:GIT_LFS_SKIP_SMUDGE="1"` [on Windows]
      `git clone -b ndbi021 https://github.com/pdokoupil/EasyStudy.git"` [on Windows]
    - `GIT_LFS_SKIP_SMUDGE=1 git clone -b ndbi021 https://github.com/pdokoupil/EasyStudy.git` [on Mac/Linux]
  - Download and **extract** `*.zip` files (images, and MovieLens dataset)
    - `ml_latest_img.zip` contains img folder, `ml-latest.zip` contains `ml-latest/*.csv` files.
      Expected structure is `server/static/datasets/ml-latest/*.csv` and `server/static/datasets/ml-latest/img/*.jpg`
  - **Alternative 1) Local installation:** Install needed dependencies in Python venv
    - `python -m venv easystudy_venv`
      - `./easystudy_venv/Scripts/activate` [on Windows]
      - `source ./easystudy_venv/bin/activate` [on Mac/Linux]
    - `pip install --upgrade pip` or `python -m pip install --upgrade pip`
    - `cd ./EasyStudy/server`
    - `pip install -r ./local_requirements.txt` (should support Python versions 3.9.x, 3.10.x, 3.11.x, 3.12.x, was tested with 3.9.6, 3.10.11, 3.11.8, 3.12.2) -> **! Specifically, Python >= 3.13 should use `local_requirements_py3_13.txt` !**
    - `flask --debug run`
  - **Alternative 2) Use docker**:
    - Build docker image (*`build_server_container.sh`*) then run docker container (*`start_server.sh`*)
  - Administration is available at http://localhost:5000/administration where you have to register an account
  - Details in README

Creating a Study

- To verify the installation works as expected, create a first study
- Visit http://127.0.0.1:5000/administration
- “Go to signup page” and create account
- “Submit”, then “Login” with the newly created account
- In “Available templates” there will be “fastcompare” plugin, use the “Create” button
- Fill-in the configuration (see this example) and then “Create” the study
- Wait until yellow background disappears indicating the study is “Initialized”
- Use “Link” under “Join Url” to open the study



Available templates

Name	Description	Version	Author	Create
fastcompare	Fast and easy comparison of 2 or 3 RS algorithms on implicit feedback datasets.	0.1.0	Patrik Dokoupil	Create

Existing user studies

Active	Creator	Error	Guid	Id	Initialized	Join Url	Parent Plugin	Participants	Settings	Time Created
TRUE Deactivate Delete	b@b.b		0lehjCH9LQp6FZKtNxA-rin5zhlrevDe	2	TRUE	Link	fastcompare	0	{"k": 10, "prolific_code": null, "n_algorithms_to_compare": 2, "result_layout": "columns", "shuffle_algorithms": true, "shuffle_recommendations": true, "algorithm_parameters": [{"l2": 0.1, "positive_threshold": 2.5, "displayed_name": "EASE A", "name": "EASE"}, {"l2": 0.1, "positive_threshold": 2.5, "displayed_name": "EASE B", "name": "EASE"}], "n_iterations": 5, "selected_algorithms": ["EASE", "EASE"], "selected_preference_elicitation": "Popularity Sampling", "preference_elicitation_parameters": {"n_samples": 10, "k": 1}, "selected_data_loader": "Filtered ML-25M dataset", "data_loader_parameters": {}, "text_overrides": {"footer": ""}, "show_final_statistics": true}	Wed, 19 Mar 2025 22:37:36 GMT

Database

- All data is stored in SQLite database (`server/instance/db.sqlite`)
  - Different tools available for viewing the data:
    - https://github.com/coleifer/sqlite-web Python package providing web-based browser (allows exporting the data as JSON/CSV)
    - https://sqlitebrowser.org/ (Windows, Mac & Linux supported)
    - SQLite Viewer and other VS Code extensions
- Tables of interest:
  - `userstudy`: records for individual user studies
  - `participation`: records for user study participants
  - `interaction`: different interactions from the users
    - e.g. elicitation-started, elicitation-ended, selected-item, …
    - filter by interaction_type, use json to load the data column
- Loading data from Python: if you use sqlite-web, you can open e.g. `interaction` click on Query then Export JSON
  - `df_interaction = pd.read_json('interaction-export.json', encoding='utf-8')`
  - You can use `json.loads` on individual `df_interaction.data` entries (e.g. `json.loads(df_interaction.iloc[0].data)`)

Extending the fastcompare plugin

- Adding new algorithms is fairly easy
  - It is enough to add implementation of the `AlgorithmBase` base
    - See `algorithm_base.py` for interface definition
    - Also see `ease.py` for an inspiration
  - Details in README

Try yourself

- Create `MostPopularPerCategory` algorithm
  - Generating recommendation of length `k`:
    - If `k <= |categories|` then sample `k` unique categories at random and take most popular item for each of them
    - Otherwise sample with replacement and do the same
    - **Use ML dataset**
    - Tips: `loader.get_all_categories`
      - `loader.get_item_index_categories`
      - `popularity[i] = |users who rated i| / |users|`

Note on debugging / testing

- Is not really convenient, usually you learn about issue only once code execution gets there
  - If your algorithm fails in fit, you have to create new study because initialization is only triggered after study creation
  - If it fails in predict, it is enough to start user study from beginning
- Issues are typically reported in terminal window / error_logfile.txt (when running in docker).
- You can add arbitrary logging via standard Python **`print`** function
- ***UPDATE** starting from 3551522 basic functionality can be tested by simple “unit test” (i.e. you do not have to create new study and you save some disk space on cache).
  - The tests are in `test_algorithm.py` which can be executed via `run_tests.sh` (or just run `pytest` from `./server`)
    - Covers verification of the most basic functionality - `fit()` does not fail, `predict()` returns right number of items and item filtering works as expected.
    - Adding new algorithm to tests can be done by extending `tested_algorithm_combinations`

Troubleshooting common issues 1

- If you cannot checkout **ndbi021** branch, ensure your git working tree is empty (i.e. no files got modified)
  - There are multiple reasons why the files could be modified right after cloning the repository, the most probable culprit is Git behavior when it comes to replacing LF with CRLF (on Windows)
    - Can be disabled by `git config --global core.autocrlf false` and then cloning the repository (or just restore the git repo to a clean state if you don't like changing your global git config)

- High memory consumption
  - Each instance of user study (created via Administration) has its own cache
    - Located in `server/cache/fastcompare/${STUDY_ID}`
    - ***UPDATE** Starting from 0d85fd9 the cache is removed upon study removal (from administration). In older versions this has to be done by manually removing per-user-study cache folders

Troubleshooting common issues 2

- If you create a new study and it does not get initialized (yellow background transitioning to white background) within ~10 minutes, there is most likely some issue
  - Data placed at wrong location (see “Setup” slide for where .csv files and img folder should be placed)
  - Runtime issue with packages (rare, but not impossible)
  - In any case, check logs for more details (console window output if running locally, or `error_logfile.txt` if running in Docker)
  - ***UPDATE** Starting from ffc29e5 if there is error in initialization, study background turns red and such study can be removed.
- Newly added algorithm does not appear in study creation config
  - Ensure you have saved the changes and restarted the server (manual restart is needed when running in Docker)
  - Ensure all methods from `AlgorithmBase` are implemented in your class
  - Ensure class name of new algorithm is unique and matches return value of `name()` method
  - Ensure you place the class at right place (e.g. create new file next to `ease.py`)

OLD Content

OLD Content

OLD Content

OLD Content