UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Big Data Project
Movie Recommender System

Student name: *Konstantinos Nikoletos, 7115112200022*

Course: *M111*
Semester: *Fall 2023*

Set-up and Execution

Configuration

This project made in Windows OS with Anaconda 4.14.0 and Python 3.10.0.

Install dependencies

Inside directory Movie-Recommender-System:

```
pip install .
```

Execution

Inside directory Movie-Recommender-System: [**recommender.py**]

```
python .\src\recommender.py -d 'data/ml-100k/'
                -n 5 -s dice -a content -i 1
```

also i added two optional flags:

-preprocess 1|0: if it will read from the precomputed files

-nrows NUMBER: how many rows in ml-latest files it will read/parse.

and for [**preprocess.py**]

```
python .\src\preprocess.py -d 'data/ml-100k/'
```

also i added two optional flags:

-u USER_ID: until which user id to calculate and save pre-processed data.

-nrows NUMBER: how many rows in ml-latest files it will read/parse.

1. Data

ml-latest

Reading and Parsing:

- ratings.csv
- tags.csv
- movies.csv

This dataset (ml-latest) describes 5-star (no zeros ratings) rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 33832162 ratings and 2328315 tag applications across 86537 movies.

My project can't work with the full ml-latest! This happens due to pandas dataframe pivot function, that crashes. Although, for records up to 1 million (-nrows 1000000), the app is fully functional.

ml-100k

Reading and Parsing:

- u.data
- u.item

This data set consists of 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies.

Full functionality on this dataset. Only TagBasedAlgo isn't implemented for this, as this dataset contains genres and not tags like ml-latest. The difference is that a tag is related to a rating in ml-latest, whereas genre describes a movie. So the wanted assignment functionality can't be reproduced with ml-100k.

2. Implementation

Data Loading

All parsers have been implemented. [recommender.py & preprocess.py]

Recommender System [recommender.py]

Has been implemented. Also each of the similarity metrics has been implemented as wanted. [metrics.py]

User-User recommendation [algorithms.py]

Implemented as described in the lectures.
It is the most expensive in time algorithm.

Item-Item recommendation [algorithms.py]

Implemented as described in the lectures.

Tag-based recommendation [algorithms.py]

Implemented as described in the lectures. Works only with the ml-latest dataset.

Content-based recommendation [algorithms.py]

Implemented as described.

Hybrid recommendation [algorithms.py]

It's a linear combination of UserUser and ItemItem algorithm joined with Content-Based algorithm, using weights. It's set with 0.5 from UserUser and 0.5 from ItemItem. The new ratings are then sorted again and top-n divided by 2 are saved. For each movie in the movies recommended, top-n divided by 2 in number, a new movie is recommended and added in the resulting recommendations. In other words for each movie in the half of the linear conjunction of UserUser and ItemItem, I get the all the recommendations from ContentBased (providing each movie id) and if the movie is not in the recommendations I add it, until top n is reached. This algorithm is really time consuming, however it both produces personalized recommendations and user-independent (ContentBased).

Real system [preprocess.py]

It runs all combinations for users and saves only for User-User and Item-Item the N most similar users-items. Saves all the N data, in pickle files under directory **preprocessed_data_100k** with an example name: **ml-100k_item_3_pearson.pkl**;ml-100k is the dataset, item the algorithm, 3 is the user id, and pearson the similarity metric.

Also, there is a directory named *preprocessed_data_latest* that contains for User 1 and User 2 of ml latest preprocessed User N sets. These N sets produced with **-nrows 1000000**:

- mllatest_user_1_cosine.pkl
- mllatest_user_2_cosine.pkl

could't save more, I run out of disk memory. This optimization reduced almost 50% the runtime.

However, nor the execution time, nor I had available disk in my PC for all users, so I have only pre-computed the ml-100k, users with ids {1 to 10}. You can check the difference by running without using the precomputed:

```
python .\src\recommender.py -d 'data/ml-100k/'
        -n 5 -s jaccard -a user -i 1
```

and running with the precomputed:

```
python .\src\recommender.py -d 'data/ml-100k/'
        -n 5 -s jaccard -a user -i 1
            -preprocess 1
```

Caution: Add flag **-preprocess 1** to utilize pre-calculated data

3. Project Challenges & Possible Solutions

Due to the complexity of the algorithms (mostly User-User and Item-Item) and the data volume, this projects has both time and memory challenges. For tackling memory issues I used sparse matrices, and python packages like numpy, which are optimized for this.
What I would do to tackle these problems if it was to become a real recommender system?

- Use **multi-processing or multi-threading** at the step that calculates for each user the recommendation. (time optimization)
- Use **distributed systems**, like Spark for data processing. This way I would avoid the crush in pandas pivot.
- Use **cache memories** for similarity metric scores.
- Choose another **programming language** (python is known for being slow).
- Re-implement with Cython all similarity metrics.

4. General remarks

I had many issues in developing this project as my laptop has 8GB of RAM and almost 60GB available on disk, so I had limited resources for experimenting.

Also, I have used some python packages, for utilities, like numpy. As in the assignment it clearly states "In other words, you cannot use libraries that directly implement one of the recommendation algorithms below". All algorithms and recommendation functionality has been implemented from scratch and none package has been used that directly provides any algorithmic functionality.

5. Deliverable structure

- data
  - ml-100k (empty)
  - ml-latest (empty)
- preprocessed_data_100k (pre calculated data for ml-100k)
- preprocessed_data_latest (pre calculated data for ml-latest)
- src (all source code)
  - movie_recommender_system
    - __init__.py
    - algorithms.py (all algos code)
    - metrics.py (all similarity metrics code)
    - utils.py (utility methods code)
  - preprocess.py (preprocess app)
  - recommender.py (recommender app)
- pyproject.toml (builds project)
- setup.py
- REAMDE (for GitHub)
- REPORT.pdf