Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

TCC/TSI/TECC: Sistemas de Recomendação

Research Challenge #2
Movie Recommendation

Deadline: Jan 6th, 2025 23:59 UTC-3 via Moodle and Kaggle

Overview The goal of this challenge is to implement a content-based movie recommender by exploiting movie metadata obtained from OMDB.[1] **Note that only personalized recommender implementations are acceptable.** In contrast to RC1, which provided a restricted setup to encourage the development of very basic collaborative recommenders from scratch, RC2 provides a rich setup to foster the research and development of multi-dimensional recommenders capable of exploring both user-item interactions as well as item content information. As discussed in class, various implementation choices impact the quality of content-based recommendations, including choices for content representation (e.g., unigrams, n-grams, concepts, named entities, latent topics), user profiling (e.g., by aggregating positive and negative feedback), and user-item similarity estimation. As part of this assignment, you should try different instantiations of these components, and verify the resulting recommendation performance of your implementation by submitting your produced recommendations to Kaggle.[2]

Kaggle This assignment uses Kaggle Community as a platform for automatically evaluating the effectiveness of your produced recommendations. You can register to join the Kaggle competition by clicking on the following link:

`https://www.kaggle.com/t/fb8028083c0544ca8e64fbc82a5698c2`

Teams This challenge can be pursued in **teams of one or two students**. Regardless, every student must register individually to join the competition using their **matriculation number** as a "Team Name". Team mergers[3] may occur before the **merger deadline on Dec 23rd, 2024 23:59 UTC-3**.

[1] `https://www.omdbapi.com/`
[2] `https://www.kaggle.com/`
[3] https://www.kaggle.com/docs/competitions#forming-a-team

Implementation You must use Python 3 for this assignment. Different from RC1, **you are free to use whatever Python library you wish** (including existing recommender system toolkits[4]) to help develop your solutions for the challenge. **Your code must run in a virtual environment and include a `requirements.txt` file with all dependencies needed to run it.** Execution errors due to missing libraries or incompatible library versions will result in a zero grade. You can test your setup in one of the Linux machines provided by the Department of Computer Science[5] using the following commands:

```
$ python3 -m venv rc2
$ source rc2/bin/activate
$ pip3 install -r /path/to/requirements.txt
```

Execution **Your implementation must include a `main.py` file**, which will be executed in the same virtual environment described above as follows:

```
$ python3 main.py ratings.jsonl content.jsonl targets.csv
```

Input Your implementation must take the following files as input:

- `ratings.jsonl`, containing 659,720 historical user-item interactions
- `content.jsonl`, containing metadata information for 38,012 movies
- `targets.csv`, containing 100 movies to be ranked for each of 6,162 users

Note that `targets.csv` contains a header line. All these files can be downloaded from the data description page on Kaggle.[6]

Output For each unique user in the `targets.csv` input file, an unordered set of 100 movies is also included. Your goal is to rank this set of movies in decreasing order of their relevance to the user. To this end, you can leverage the historical user-item matrix available from the `ratings.jsonl` file as well as the movie metadata available from the `content.jsonl` file. The produced top-100 rankings for each of the 6,162 test users must be written to standard output.[7]

Submissions The rankings output by your implementation must be stored in a submission (csv) file to be uploaded to Kaggle. In total, a submission file must contain a header line plus one line for each of the $6,162 \times 100 = 616,200$ $\langle user, item \rangle$ pairs to be ranked (i.e. a submission file must contain a total of 616,201 lines). An example submission file is provided next:

[4] `https://recommender-systems.com/resources/software-libraries/`
[5] `https://www.crc.dcc.ufmg.br/doku.php/infraestrutura/laboratorios/linux`
[6] `https://www.kaggle.com/c/recsys-20242-rc2/data`
[7] `https://en.wikipedia.org/wiki/Standard_streams`

```
UserId,ItemId
0006246bee,80d1dae630
0006246bee,aad36aac60
0006246bee,c1ee6829f5
...
001fca6b30,6fdd8f48f6
001fca6b30,7ececef73a
001fca6b30,9466eed1c5
...
fffffe98d0,274facc0c2
fffffe98d0,2171a9b7a8
fffffe98d0,3200815a04
```

Your submission should be uploaded to Kaggle[8] to be automatically evaluated. Through the course of this challenge, you should try multiple instantiations of the various components of your implemented recommender, in the hope of further improving its effectiveness. To this end, you can upload a maximum of 20 submissions per day to Kaggle. The platform will maintain a live leaderboard indicating the relative performance of your submissions in comparison to those made by your fellow classmates. Keep track of the performance of your submissions, so you can analyze what worked in your final presentation.

Deliverables Before the deadline (Jan 6th, 2025 23:59 UTC-3), you must submit a package file (`zip`) via Moodle containing the following:

1. Source code (including `main.py` and `requirements.txt`);
2. The best submission (csv) uploaded to Kaggle.

Grading policy This assignment is worth a total of 20 points, with the possibility of attaining up to 3 extra points. These points are distributed as:

- 5 points for your *presentation* to be delivered online on Jan 8th, 2025, describing the strategies you explored throughout the challenge (ideally with accompanying illustrations), along with an analysis of the effectiveness of these strategies (what worked, what did not work, noting the performance achieved on the leaderboard with each strategy). It is not necessary to describe implementation details or the computational complexity of each investigated idea. As previously discussed, RC2 focuses less on implementation and more on the research behind the solutions that led to your best result on Kaggle. It is also worth highlighting if any existing recommendation toolkit was used, which datasets were explored (among those provided or even external data), how these datasets were processed, which recommendation algorithms were used, and how they were trained. **Unlike RC1, there will be no separate report for RC2; instead, your presentation will serve as a report.**

[8] `https://www.kaggle.com/c/recsys-20242-rc2/submissions`

- 10 points for your *performance*[9] relative to the provided non-personalized benchmarks on Kaggle[10] (1 point for outperforming each of the random, most-popular, best-rated, etc. benchmarks).
- 5 points for your *performance* relative to your fellow contestants.[11]

To be eligible for the performance grades, you must satisfy the following:

1. You must upload at least one submission to Kaggle within the timeframe of this assignment;
2. The source code that you submit (via Moodle) by the deadline should be able to precisely generate your last submission to Kaggle;
3. Your implementation should be able to execute correctly in a Linux environment under 60 minutes.

[9] Performance will be assessed based on the nDCG@20 score of your last submission on Kaggle's private leaderboard, which should closely resemble your performance on Kaggle's public leaderboard provided that your solution does not overfit.

[10] `https://www.kaggle.com/c/recsys-20242-rc2/leaderboard`

[11] In a nutshell: stay within the average of all contestants, and you get roughly all 5 awarded points; try your best to outperform the average, and you get up to 3 extra points.