Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

TCC/TSI/TECC: Sistemas de Recomendação

Programming Assignment #2
Content-based Movie Recommendation

Deadline: May 14th, 2018 23:59 UTC-3 via Moodle and Kaggle

Overview The goal of this assignment is to implement a content-based movie recommender by exploiting movie metadata obtained from OMDB.[1] 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]

Teams This assignment should be performed individually. Any sign of plagiarism will be investigated and reported to the appropriate authorities.

Source code Your implementation should be produced in C or C++. You **must not** use any external library other than the C or C++ standard libraries.[3][4] You should provide a `Makefile`[5] for compilation and execution.

Compilation Your source code will be compiled in a modern Linux environment (Ubuntu 14.04.3 LTS). The compilation **must not** produce any warnings or errors. To make sure your compilation succeeds, you can test it in one of the Linux machines provided by the Department of Computer Science.[6]

[1] `http://omdbapi.com/`

[2] `http://www.kaggle.com`

[3] `https://en.wikipedia.org/wiki/C_standard_library`

[4] `https://en.wikipedia.org/wiki/C++_Standard_Library`

[5] `https://en.wikipedia.org/wiki/Makefile`

[6] `http://www.crc.dcc.ufmg.br/infraestrutura/laboratorios/labs_unix`

Execution Your `Makefile` should include the following target named `run`, which will be used when executing your compiled program:

```
./recommender content.csv ratings.csv targets.csv > submission.csv
```

Input As shown above, your executable will take three CSV files as input:

- `content.csv`, containing 22,080 $\langle item, content\rangle$ pairs[7]
- `ratings.csv`, containing 336,672 $\langle user, item, rating\rangle$ historical tuples
- `targets.csv`, containing 77,276 $\langle user, item\rangle$ pairs for prediction

Note that each of these input files contains a header line. These files can be downloaded from the data description page in Kaggle.[8]

Output For each of the $\langle user, item\rangle$ pairs in `targets.csv`, your implementation should predict the corresponding numeric rating, by leveraging the item metadata available from `content.csv` and the historical user-item matrix available from `ratings.csv`. You **must not** use any historical information about the target item. The resulting prediction should be written to standard output[9] as a $\langle user, item, rating\rangle$ tuple, formatted as two CSV columns:

- `UserId:ItemId`, containing the $\langle user, item\rangle$ pair separated by a colon (:)
- `Prediction`, containing the predicted rating for the target pair

Submissions The predictions output by your executable should be written to a submission file, called `submission.csv`. In total, a submission file must contain a header line plus one line for each of the n predictions. For this assignment, n = 77,276, meaning that your submission should have $n + 1$ = 77,277 lines. An example submission file is provided below:

```
UserId:ItemId,Prediction
u0000039:i0060196,6.38666836618544
u0000039:i0099077,3.10922975975217
u0000039:i0102926,8.02790645781112
...
u0038723:i1951265,2.98912620664908
u0038723:i2395427,6.44312165951174
u0038723:i2413496,0.13009045472507
```

[7] The `content` column is formatted as a JSON document. You are allowed to use an external library for JSON parsing for C/C++ from the ones listed at `http://www.json.org/`.

[8] `https://www.kaggle.com/c/recsys-20181-cbmr/data`

[9] `https://en.wikipedia.org/wiki/Standard_streams`

Your submission file should be uploaded to Kaggle[10] in order to be automatically evaluated. Throughout the course of this assignment, you should try alternative instantiations of the various components of your implemented recommender, in the hope of further improving the quality of your produced recommendations. To this end, you can upload a maximum of 10 submissions per day to Kaggle. The platform will maintain a live leaderboard indicating the relative performance of your generated recommendations in comparison to those generated by your fellow classmates.

Kaggle This assignment uses Kaggle in Class as a platform for automatically evaluating the quality of your produced recommendations. If you do not yet have a Kaggle account, you can register by clicking on the following link:

`https://www.kaggle.com/t/7e9e47d858ec4e37a21da486ee4249ee`

Make sure to use your **matriculation number** as your "Team Name".

Evaluation The evaluation metric for this assignment is Root Mean Squared Error (RMSE).[11] The RMSE score, commonly used in the recommender systems literature, measures accuracy by penalizing prediction errors according to:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{1}$$

where y_i and $\hat{y}_i$ are respectively the expected and predicted values for the i-th $\langle \textit{user}, \textit{item} \rangle$ pair of a total of n predictions made.

Grading policy This assignment is worth a total of 15 points, with the possibility of attaining up to 5 extra points. These points are distributed as:

- 5 points for your *documentation*, assessed based on a short (pdf) report[12] describing your implemented data structures and algorithms, their computational complexity, as well as a discussion of your attained results (e.g., based on the submissions you uploaded to Kaggle).

- 5 points for your *implementation*, assessed based on the quality of your source code, including its overall organization (modularity, readability, indentation, use of comments) and appropriate use of data structures.

- 5 points for your *performance*, assessed based on the RMSE score of your last submission uploaded to Kaggle. In particular, your final grade for this

[10] `https://www.kaggle.com/c/recsys-20181-cbmr/submissions`

[11] `https://www.kaggle.com/wiki/RootMeanSquaredError`

[12] Your documentation should be no longer than 4 pages and use the ACM LaTeX template (sample-sigconf.tex): `https://www.acm.org/binaries/content/assets/publications/consolidated-tex-template/acmart-master.zip`

criterion will depend upon your performance relative to the performance of the other contestants in Kaggle's private leaderboard.[13] Let e denote the set of all contestants' RMSE scores in the final private leaderboard. The grade g_i of the i-th contestant will be computed as follows:

$$g_i = 5 - 5z_i \tag{2}$$

$$z_i = \frac{s_i - \mu_s}{\sigma_s} \tag{3}$$

$$s_i = \frac{e_i - \min(e)}{\max(e) - \min(e)}. \tag{4}$$

While your performance is originally worth 5 points, the final grade g_i will lie in the interval [0,10], i.e., you could have up to 5 extra points depending upon your performance. In order to avoid trivially awarding 0 to the worst submission and 10 to the best submission, we define $\min(e)$ and $\max(e)$ above according to:

$$\min(e) = \mu_e - 3\sigma_e \tag{5}$$

$$\max(e) = \mu_e + 3\sigma_e. \tag{6}$$

In a nutshell, g_i could take the following values:

$$0, \text{ if } e_i \to \mu_e + 3\sigma_e; \tag{7}$$

$$5, \text{ if } e_i \to \mu_e; \tag{8}$$

$$10, \text{ if } e_i \to \mu_e - 3\sigma_e. \tag{9}$$

In plain English: stay within the average, and you get roughly all 5 awarded points; try your best to outperform the average, and you get up to 5 extra points. In order to be eligible for the performance grades, you must satisfy the following three criteria:

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. The program compiled from your source code should be able to execute correctly in a Linux environment under 5 minutes.

Deliverables Before the deadline (May 14th, 2018 23:59 UTC-3), you should submit a package file (zip or tar.gz) via Moodle containing the following:

1. `Makefile` (see **Compilation** and **Execution** above);

[13] Your performance on Kaggle's public leaderboard will normally reflect your performance on the private leaderboard, provided that your solution does not overfit.

2. Source code and corresponding executable binary;
3. The last submission file (csv) uploaded to Kaggle;
4. Documentation file (pdf).