COMP47580
Recommender Systems & Collective Intelligence

Recommender Systems Assignment – Part 2

Introduction

This assignment involves implementation and running experiments. For this, the second part of the RS assignment, an item-based collaborative filtering (IBCF) recommender framework is provided for you to extend. Download the framework (an Eclipse project) from Moodle and import it into Eclipse.

Notes

- Submission deadline:
    - Submit your code and graphs for this part of the assignment by **Thursday, 7th March, 23:59 hrs**.
    - See section **Submission Instructions** for more details on what needs to be submitted.
- Late submissions policy – when coursework is submitted late the following penalties apply:
    - Coursework submitted at any time up to one week after the due date will have the grade awarded reduced by two grade points (for example, from B- to C) or, for assessment marked as a percentage, the mark reduced by 10 (for example, from 87% to 77%).
    - Coursework submitted more than one week but up to two weeks after the due date will have the grade reduced by four grade points (for example, from B- to D+) or, for assessment marked as a percentage, the mark reduced by 20 (for example, from 87% to 67%).
    - Coursework received more than two weeks after the due date will not be accepted.

- **Important** – this is not a team/group assignment – each student must submit her/his own work. Please ask if you have any questions about this. See the course Moodle for information on the UCD plagiarism policy.

<u>**Implementation**</u>

Implement the following:

1. Similarity computation:
    - Implement the mean squared difference similarity metric.
        - Implement this similarity metric in a new class named `MeanSquaredDifferenceMetric` in package `similarity.metric`. Ensure the class implements the `SimilarityMetric` interface (in package `similarity.metric`).
    - Implement the Cosine similarity metric.
        - Implement this similarity metric in a new class named `CosineMetric` in package `similarity.metric`. Ensure the class implements the `SimilarityMetric` interface (in package `similarity.metric`).

2. Neighbourhood formation:
    - Implement the threshold neighbourhood approach.
        - Implement this neighbourhood approach in a new class named `ThresholdNeighbourhood` in package `alg.ib.neighbourhood`. Ensure the class extends the abstract `Neighbourhood` class (in package `alg.ib.neighbourhood`).

3. Prediction generation:
    - Implement the weighted average predictor.
        - Implement this predictor in a new class named `WeightedAveragePredictor` in package `alg.ib.predictor`. Ensure the class implements the `Predictor` interface (in package `alg.ib.predictor`).
    - Implement the deviation from item-mean predictor.
        - Implement this predictor in a new class named `DeviationFromItemMeanPredictor` in package `alg.ib.predictor`. Ensure the class implements the `Predictor` interface (in package `alg.ib.predictor`).

Experiments

Perform the following experiments:

1. Effect of neighbourhood size on predictions:
    - Using the nearest neighbourhood approach, plot overall RMSE and coverage versus neighbourhood size for each of the four predictors: non-personalised (provided in the framework), simple average (provided in the framework), weighted average, and deviation from item-mean.
    - Use neighbourhood sizes of 10, 20, 30, …, 250.
    - Use Cosine similarity in this experiment.
    - Plot two graphs:
        - RMSE versus neighbourhood size (show performance for the four predictors on one graph).
        - Coverage versus neighbourhood size (show performance for the four predictors on one graph).

2. Effect of neighbourhood threshold on predictions:
    - Using the threshold neighbourhood approach, plot overall RMSE and coverage versus threshold for the deviation from item-mean predictor only.
    - Use thresholds of 0, 0.05, 0.10, 0.15, …, 0.70.
    - Use Cosine similarity in this experiment.
    - Plot one graph:
        - Show RMSE and coverage versus threshold on one graph.

3. Effect of similarity metric on predictions:
    - Using the deviation from item-mean predictor only, compare the overall RMSE and coverage achieved by the following similarity metrics: Cosine, Pearson correlation, Pearson correlation with significance weighting (N = 50), and mean squared difference.
    - Use the nearest neighbourhood approach with a neighbourhood size of 200 in this experiment.
    - Plot one graph:
        - Show RMSE and coverage versus similarity metric on one (e.g. clustered column) graph.

Note: in the above, coverage measures the percentage of test set ratings for which predictions that can be made.

Submission Instructions

Submit all your code and graphs by the due date. All submissions are to be made via Moodle only:

- Create a **zip file** of your exported Eclipse project using the following filename: `IBCF_12345678_Code.zip` (where `12345678` is your student number).

- In the Eclipse project export, include:
  - All code provided in the framework and the code you implemented yourself.
  - Your graphs – save these in folder `graphs` in the Eclipse project.
  - A **task guide document** that includes the following:
    - Include clear instructions on how to execute your code for each of the experiments. The best approach is to create a copy of the `ExecuteIB_ML20M.java` class (appropriately named, e.g. `ExecuteIB_Expt_X`) in package `alg.ib` for each experiment or part of experiment.
    - Save the task guide document in folder `task_guide` in the Eclipse project.

Notes:

- **When submitting your code (an export of your Eclipse project), please do not edit any of the classes provided in the framework. The export of your Eclipse project should include all the classes provided (unedited) and the classes you have created yourself.**

- *If you do not follow these instructions, your code may not run or may output incorrect results. If so, you will not be awarded any marks for your code.*

- Of course please feel free to experiment with the framework – but do this using a local copy of the framework, and please remember to submit an export of the correct version (i.e. unedited, and including the classes you have created) of the framework to Moodle.