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, a user-based collaborative filtering (UBCF) 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, 5th March, 23:59 hrs**.
  - See section **Submission Instructions** for 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 and including 5 working days after the due date will have the grade awarded reduced by one grade point (for example, from B- to C+).
  - Coursework submitted more than five working days but up to and including ten working days after the due date will have the grade reduced by two grade points (for example, from B- to C).
  - Coursework received more than ten working days after the due date will not be accepted by the School.

- **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.

Implementation

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`). The constructor of the class should take two `double` arguments to specify the minimum and maximum ratings in the data set, respectively.
   - Implement the Pearson similarity metric with significance weighting.
     - Implement this similarity metric in a new class named `PearsonSigWeightingMetric` in package `similarity.metric`. Ensure the class implements the `SimilarityMetric` interface (in package `similarity.metric`). The constructor of this class should take an `int` argument that specifies the significance weighting threshold value.

2. Neighbourhood formation:
   - Implement the threshold neighbourhood approach.
     - Implement this neighbourhood approach in a new class named `ThresholdNeighbourhood` in package `alg.ub.neighbourhood`. Ensure the class extends the abstract `Neighbourhood` class (in package `alg.ub.neighbourhood`). The constructor of this class should take a `double` argument to specify the neighbour similarity threshold value. Only those users with similarity greater than the threshold value should be considered as neighbours.

3. Prediction generation:
   - Implement a simple non-personalised predictor.
     - Implement this predictor in a new class named `SimpleNonPersonalisedPredictor` in package `alg.ub.predictor`. Ensure the class implements the `Predictor` interface (in package `alg.ub.predictor`). The prediction returned should be the mean of the target item's ratings in the training set.
   - Implement the weighted average predictor.
     - Implement this predictor in a new class named `WeightedAveragePredictor` in package `alg.ub.predictor`. Ensure the class implements the `Predictor` interface (in package `alg.ub.predictor`).
   - Implement the deviation from user-mean predictor.
     - Implement this predictor in a new class named `DeviationFromUserMeanPredictor` in package `alg.ub.predictor`. Ensure the class implements the `Predictor` interface (in package `alg.ub.predictor`).

Experiments

Run 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, simple average (provided in the framework), weighted average, and deviation from user-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).

   - **Note:** to run this experiment, execute class `ExecuteUB_Expt1` in package `alg.ub`. *Please do not edit this class; if you do, your results may be incorrect and/or the format of the output may change – if so, marks will be lost.*

2. Effect of neighbourhood threshold on predictions:
   - Using the threshold neighbourhood approach, plot overall RMSE and coverage versus threshold for the deviation from user-mean predictor only.
   - Use thresholds of 0, 0.05, 0.10, …, 0.80.
   - Use Cosine similarity in this experiment.
   - Plot one graph:
     - Show RMSE and coverage versus threshold on one graph.

   - **Note:** to run this experiment, execute class `ExecuteUB_Expt2` in package `alg.ub`. *Please do not edit this class; if you do, your results may be incorrect and/or the format of the output may change – if so, marks will be lost.*

3. Effect of similarity metric on predictions:
   - Using the deviation from user-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:** to run this experiment, execute class `ExecuteUB_Expt3` in package `alg.ub`. *Please do not edit this class; if you do, your results may be incorrect and/or the format of the output may change – if so, marks will be lost.*

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: `UBCF_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.