COMP47580

Recommender Systems & Collective Intelligence

Recommender Systems Assignment – Part 1

Introduction

This part of the assignment involves the implementation of non-personalised recommender algorithms and running experiments. A Java framework is provided for you to extend. To begin, download this 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, 21st February, 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.

Non-personalised Recommender Algorithms

Implementation

Implement the following similarity metrics:

- **GenomeMetric.** Movie-movie similarity is calculated by the cosine of the angle between the movie genome scores vectors. Here, two movies are considered similar if both movies have similar scores for each genome.
- **RatingMetric.** Movie-movie similarity is calculated by the cosine of the angle between the movie rating vectors. Here, two movies are considered similar if both movies have similar ratings patterns. For this similarity metric only, if a user has not rated a movie, assume the rating is zero. Also, if none of the users have rated both movies, the similarity should be set to zero.
- **IncConfidenceMetric.** Movie-movie similarity is calculated by the product association approach; in particular, similarity is given by the *increase in liking Y if X is liked* (see lecture notes). The following approach should be used.

  For simplicity, assume X and Y are individual movies and a movie is considered liked by a user if the assigned rating is ≥ *t* (**where *t* = 4.0**). In what follows, N denotes the total number of users in the dataset.

  supp(X) – the percentage of users which have assigned a rating ≥ *t* to X:

  = (# users which have assigned a rating ≥ *t* to X) / N

  supp(X and Y) – the percentage of users which have assigned a rating ≥ *t* to both X and Y:

  = (# users which have assigned a rating ≥ *t* to both X and Y) / N

  conf(X => Y) is given by: supp(X and Y) / supp(X)

  supp(!X) – the percentage of users which have assigned a rating < *t* to X:

  = (# users which have assigned a rating < *t* to X) / N

  supp(!X and Y) – the percentage of users which have assigned a rating < *t* to X and a rating ≥ *t* to Y:

  = (# users which have assigned a rating < *t* to X and a rating ≥ *t* to Y) / N

  conf(!X => Y) is given by: supp(!X and Y) / supp(!X).

  The **similarity** between movies X and Y is then given by:
  conf(X => Y) / conf(!X => Y).

It is important to check for division by zero in the following cases:

- conf(X => Y) = supp(X and Y) / supp(X)
- conf(!X => Y) = supp(!X and Y) / supp(!X)
- similarity = conf(X => Y) / conf(!X => Y)

In any of the above, if the denominator is equal to zero, the result of the division should be set to zero; otherwise the output of the method will be incorrect.

The same is true for the other similarity metrics. For example, see how division by zero is handled in method `getItemSimilarity` in the `GenreMetric` class.

Separate classes for each of these three similarity metrics have been created in package `alg.np.similarity.metric`. Each class contains the following method to calculate the similarity between two movies (where `X` and `Y` are the ids of the first and second movies, respectively):

```
public double getItemSimilarity(final Integer X, final Integer Y)
```

Currently, this method in each class simply returns a similarity value of zero. The task then is to implement this method in each class according to the approaches described above.

Notes:

- An implementation of the genre similarity metric is provided in the framework (class `GenreMetric` in package `alg.np.similarity.metric`). In this case, the similarity between two movies is given by the Jaccard index calculated over the set of genres associated with each movie.
- **When submitting your code (an export of your Eclipse project), please do not edit any of the classes provided in the framework (except, of course, for the three similarity metric classes):**
    - When grading your code, classes `ExecuteNP` and `ExecuteNP_Expt` will be run (these classes are provided in the framework in package `alg.np`).
    - *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 except for the three similarity metric classes) of the framework to Moodle.

Experiments

Once the similarity metrics have been implemented, run the class `ExecuteNP` (in package `alg.np`). This class will output the top-3 most similar movies for a sample set of five target movies. Currently, this class is configured to run using the `GenreMetric` (in package `alg.np.similarity.metric`) similarity metric. To run class `ExecuteNP` using a different similarity metric (e.g. `GenomeMetric`), change line #32 (in class `ExecuteNP`) from:

```
SimilarityMetric metric = new GenreMetric(reader);
```

to:

```
SimilarityMetric metric = new GenomeMetric(reader);
```

Run the class using each of the four similarity metrics and note the difference between the top-3 recommendations that are made for each of the target movies.

Next, run the class `ExecuteNP_Expt` (in package `alg.np`). This class will output the following statistics for the non-personalised algorithm using each of the four similarity metrics (`GenreMetric`, `GenomeMetric`, `RatingMetric`, `IncConfidenceMetric`):

Recommendation relevance:

- The relevance of a recommended movie is given by the mean of the ratings the movie received in the training set.
- For a given target movie, the relevance of the top-k recommendations made is calculated by taking the average of the mean rating over each recommended movie.
- The statistic reported is the average over all target movies.

Coverage is given by the percentage of target movies for which at least one recommendation can be made.

Recommendation coverage is given by the percentage of movies in the dataset which appear at least once in the top-k recommendations made over all target movies.

Item-space coverage:

- For a given target movie, the percentage of movies in the system which are capable of being recommended (i.e. those movies which have a similarity greater than zero with the target movie) is calculated.
- The statistic reported is the average percentage over all target movies.

Recommendation popularity:

- The popularity of a recommended movie is given by the percentage of users in the system which have rated the movie.
- For a given target movie, the popularity of the top-k recommendations made is calculated by taking the average of the popularity over each recommended movie.
- The statistic reported is the average over all target movies.

Plot two graphs:

- In the first graph, show recommendation relevance versus similarity metric.
- The second graph should show coverage, recommendation coverage, item-space coverage, and recommendation popularity versus similarity metric all in one graph (for example, use a clustered column graph in MS Excel).

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: `NP_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.