COMP3208 Social Computing Techniques

Muhammad Burhan Hafez, Burhan.hafez@soton.ac.uk

Updated: 27th January 2025

Deliverables and deadlines

Assignment Number	Deliverable(s)	Deadline	Marking Scheme
1	source_code.txt results.csv	Friday, 14 March 2025 @ 16:00 Module week 7	15 Marks (results.csv)
2	source_code.txt results.csv	Friday, 9 May 2025 @ 16:00 Module week 11	10 Marks (results.csv) 15Marks (source_code.txt)

Task

The aim of the coursework is to build and evaluate realistic recommender system algorithms. This is broken down into two assignments, each testing a different element of the overall goal. Individual submissions are expected for this coursework (i.e. not group work).

Each of the two assignments expects the same deliverable format to be submitted via ECS handin. There should be a source_code.txt file and a results.csv file in each submission.

You should run your source_code offline (on your laptop or ECS hardware) and only submit it when it's ready for evaluation. Each task expects you to calculate a set of results which are then submitted for automated evaluation alongside the source code itself.

The source_code used to run your code must be a single file in either .java or .py format (renamed as source_code.txt). You cannot submit files in other formats, such as Eclipse project files or Jupyter notebook .ipynb files. Multiple file submissions are not allowed, there must be only a source_code.txt file and a results.csv file in each submission.

Your source code must only use in-built java or python libraries. The only exception to this rule is for the python lib numpy which is allowed to make array manipulation easier. For example, scipy and scikit_learn libs are not built into python and so are not permitted. Same goes for weka java libs, these are not part of java and so are not permitted. The assignment is intended to evaluate your ability to understand and develop recommender system algorithms from scratch, not your ability to use powerful third-party libraries to do the task efficiently without a deep understanding.

Assignments are evaluated automatically via the ECS handin system. For each assignment, 10 submission attempts are allowed (formative), with the best scoring attempt used for the final mark (summative).

The source_code must be self-described, using easy-to-read inline comments to explain how the coded algorithm works. Your self-described code should provide a sufficiently detailed explanation for how each algorithm works (e.g. narrative to code steps alongside an explanation of maths behind the algorithm) to provide evidence of a deep understanding of the algorithms used. Use your

judgement for how long the inline comments should be, providing enough information to show deep understanding but not so much it becomes hard to read or needlessly bloated.

An example submission is provided so the format of submission files and examples of self-described comments are clear.

For the 2nd assignment, the source_code self-described comments will be manually assessed via a code review, and marks will be provided for evidence of a deep understanding of the algorithms used and clarify of explanations. Code with no comments to explain it will be awarded zero marks in the code review.

Assignment #1 [15 marks] = small scale cosine similarity recommender system algorithm

The source_code must code a cosine similarity recommender system algorithm to train and then predict ratings for a small (100k) set of items.

Feedback >> Automated evaluation of results to compute the MAE score of predictions. To get a good mark your algorithms will need to do much better than the easiest baseline of a hardcoded recommender system which returns a fixed value (e.g. average ratings of corpus; average for a particular item; average for a particular user). As an indication, for the cohort of the last year, an MAE 0.77 was awarded an average score (~70%) and an MAE 0.701 was awarded the best score (100%). However, these values may change as a different test set will be given this time.

Marks >> Marks assigned based on MAE [15 marks]

Note: no extension will be given for Assignment 1. If there is a slight delay (with a valid reason communicated to the module leader) there will be no penalty. Otherwise, the weight of this assignment will be carried over to Assignment 2.

Optional Assignment [0 marks] = small scale matrix factorisation recommender system algorithm

The source_code must code a small-scale matrix factorisation recommender system algorithm to train and then predict ratings for a small (100k) set of items. This task is optional and does not contain any marks. The goal of this task is for the students to practise matrix factorization on a smaller dataset before using a large dataset without having to deal with database.

Marks >> 0 marks

Assignment #2 [25 marks] = large scale matrix factorisation recommender system algorithm

The source_code must code a large-scale matrix factorisation recommender system algorithm to train and then predict ratings for a large (20M) set of items. You may need to use a database to handle the large numbers of ratings.

Note: If you really have trouble with assignment 3 and cannot get your source_code to generate any predictions at all, then you can submit the self-described code anyway with an empty results file. You will score zero marks for the empty results (10), but you will still be able to score 'method' marks (15) for your self-described code if you show evidence of a deep understanding of the algorithm and maths behind it.

Feedback >> Automated evaluation of results to compute the MAE score of predictions. The MAE score will be compared against a low-scoring benchmark, which is a basic cosine similarity

recommender system algorithm without any optimisation work. To get a good mark your algorithms will need to do better than a cosine similarity algorithm. As an indication, for the cohort of the last year, an MAE 0.66 was awarded an average score (~70%) and an MAE 0.61 was awarded the best score (100%). However, these values may change as a different test set will be given this time.

Marks >> Marks assigned based on MAE [10 marks]

Marks >> Manual inspection of self-described code assessing the criteria of (a) clarity of self-described code and (b) depth of understanding of algorithm and maths behind it. Submission of an incorrect or impossible to read source code file (i.e. a file that is not a python or java source file serialized as a single plain text file) will result in a zero mark [15 marks]

Notes and Restrictions

Make sure that you provide a prediction for each of the rows in the test set. Failure to provide a complete set will result in missing predictions, which will be set to zero by default and cause a higher MAE rate than would otherwise be achieved.

Given the size of the large dataset, you may choose to use a database. A simple database such as SQLite suffices. Example java and python code for using a database is provided to help and using it will not count as plagiarism. It is not a requirement to use it, however.

Feedback

Feedback will be returned automatically for this coursework each time you submit one of your 10 submission attempts per assignment, in the form of an emailed MAE report (not marks) based on automated evaluation of your submitted rating predictions.

The final marks you get for each assignment will be emailed 4 weeks after the deadline. This allows time for marks to be computed, student extensions processed etc. This mark confirmation email will contain any additional written feedback.

Learning Outcomes

B1. Use recommender technologies such as item-based and user-based collaborative filtering techniques

D1. Set up social computing experiments and analyse the results using a scientific approach

Late submissions

Late submissions will be penalised according to the standard rules.

The handin submission time is based on your last submission, so if you submit after the deadline you will incur a late penalty.

Plagiarism

source_code will be checked using an automated code similarity checker. Do not cut and paste code from online sources like tutorials (i.e. plagiarism) or other students (i.e. collusion). Write your own code and your own self-described comments. Reusing your own work from earlier submissions for assignments in this module is explicitly allowed. Code and comment similarity checks will be on a per-assignment basis, with your best scoring source_code compared to other students best scoring source_code for that assignment.

Any violations, deliberate or otherwise, will be reported to the Academic Integrity Officer.