Web Recommender Systems 2024: Re-Exam Project

May 27, 2025

This project is composed of eight exercises. All exercises of this project are compulsory. The project will be graded as a whole. The project should be completed **individually**. You should submit:

- A **report** detailing what you have implemented, and your results and observations, for each part of the project;
- **Code** to run your experiments and **documentation** (`readme` file) on how to run it.

The format of the report must be a PDF document using the ACL template[1], no more than **7 pages** (not including references, if needed). Only content within the page limit will be graded. Change of font size is not permitted unless for tables and figures. Detailed instructions regarding the template and format can be found in Absalon.

You need to submit both the report and the code on Digital Exam: The report should be submitted as the main file and the code as a `.zip` file in the attachment. Please do **NOT** include the dataset in your `.zip` file.

The submission deadline is no later than **Friday, the 20th of June 2025 at 12:00 (noon)**.

1 Exercise 1: Familiarize Yourself with the Datasets

The purpose of the first exercise is to familiarize yourself with the programming basics needed to process the dataset that is used in the project. You will need to do statistical analysis to attain insights into the dataset.

You can find the dataset, additional supporting files, and information about them in Absalon:

`https://absalon.ku.dk/courses/80396/files/folder/reexam/data`

You will use the New Mexico partition of the Google Local Data (2021) dataset.[2] This is a dataset from Google Maps. The complete data contains roughly 5 million reviews of businesses, which are computationally expensive to

[1] `https://2021.aclweb.org/downloads/acl-ijcnlp2021-templates.zip`

[2] The complete dataset and further information about it can be found on: `https://mcauleylab.ucsd.edu/public_datasets/gdrive/googlelocal`.

process. Therefore, for this project, you will use a subset of the dataset that has been provided. The project dataset contains $\sim$ 90 000 ratings (1–5) from nearly 1200 users on $\sim$ 700 businesses.

We have provided train (*train.tsv*) and test (*test.tsv*) splits with $\sim$ 80% and $\sim$ 20% of the data respectively. These files can be found in Absalon.

In Exercises 4 and 5 (see Sections 4 and 5), we will use the metadata file *metadata.tsv*, which contains information about the businesses reviewed by the users. We will primarily use the `name` and `description` fields from *metadata.tsv*, though you are welcome to explore additional features. For each business, the metadata also includes the following fields: `category`, i.e., one or more categories describing the business (e.g., Museum, Pharmacy), `price`, that is a price indicator ranging from a single dollar sign ($) for affordable businesses to four dollar signs ($$$$) for the most expensive ones, and `address`, the physical address of the business.

In this first part, you need to:

1. Download and import the dataset splits.
2. Clean both splits from missing ratings and duplicates (cases where the same user has rated the same item[3] multiple times) if any. Sort the duplicate entries in ascending order by user id, item id, and time. Keep only the last row, i.e., the most recent rating.
3. Double check that all users from the test split also appear in the train set, and remove those that do not appear in training.
4. Compute user and item statistics (such as distribution of ratings per user/item, the top 5 most popular items) for the training set *train.tsv* and write a discussion; does the dataset have important properties that should be taken into account or that may mislead the evaluation?

2 Exercise 2: Baseline Models

In this exercise, you should implement two baseline models, TopPop and Random. Both are baseline models that will be used to benchmark the performance of other recommender models.

TopPop. This model should recommend the most popular highly-rated items based on the top-k businesses with high ratings (rating $\geq$ 3) in the training split, *train.tsv*.

Random. To each user, this model should recommend k different random items from the training split.

3 Exercise 3: Collaborative Filtering Recommender System

For this part, you can use the Python library Scikit-Surprise. Please find the documentation here: `https://surprise.readthedocs.io/en/stable/`. However, you are free to use any libraries of your choice.

[3] From now on, we will refer to businesses as items interchangeably

- Use the training set to create a user-item matrix with the ratings as the entries for rated items, and empty entries for the rest. Discuss how the user-item matrix can be built in a different way.
- Define a neighborhood-based model and a latent factor model that uses the observed entries in the user-item matrix to predict the unobserved entries. Report your choice of models.
- Use 3-fold cross-validation on the training set to tune the hyperparameters of the chosen models (similarity measure and number of neighbors for the neighborhood-based model; number of latent factors and number of epochs for the latent factor model).
- Report the hyperparameter search space. The search space can be reported in the appendix.
- Report (in a table) the optimal hyperparameters together with the corresponding validation Hit Rate (HR@k) or Mean Reciprocal Rank (MRR@k) averaged over the 3 folds.
- Run the models with the optimal hyperparameters to the whole training set.
- Use the final models to rank the unobserved (non-rated) items for each user. This ranking will be used for the evaluation part.

4 Exercise 4: Text Representation

In this part, you will work with the textual content from the dataset and will apply NLP techniques to represent businesses and users in a vector space.

1. Select the `name` and `description` fields of the businesses belonging to our train set[4]. Apply appropriate text preprocessing techniques, for example: tokenization, lowercasing, and stopword[5] removal (not necessarily in this order). Report the vocabulary size after preprocessing. There are many libraries you can use, including but not limited to, NLTK, spaCy or CoreNLP (requires Java).
2. Using the resulting text from the previous step, represent each business using pretrained word embeddings (e.g., GloVe, word2vec).
3. Explore the similarity between businesses by computing their cosine similarity. Discuss what you find.

5 Exercise 5: Content-Based Recommender System

In this week, you will continue using the train and test splits defined in Section 1.

[4]Here, you are welcome to include more features from the metadata

[5]Lists of stopwords for different languages: `http://members.unine.ch/jacques.savoy/clef/`

1. Represent each user in the same vector space that you chose in the previous exercise for representing the businesses. This can be done by averaging the representations of items that the user has rated. Note: the user representations and item representations all have the same number of dimensions.

2. Find the 10 most relevant businesses to recommend to each user. A metric such as cosine similarity or Euclidean distance can be used.

6 Exercise 6: Hybrid Recommender Systems

Create one hybrid recommender system by combining the collaborative filtering model with the content-based model using any of the following three strategies:

- *Parallel strategy* that re-ranks the items by combining their individual ratings or rank positions from the two models.
- *Switching strategy* that uses the recommendations from the collaborative filtering model for some users and the recommendations from the content-based model for other users chosen by a predefined condition.
- *Pipelining (sequential) strategy* where the output or other components of one model are used as input to the other model.

7 Exercise 7: Evaluation of Recommender Systems

In this exercise, you need to evaluate all of your models on the *test* data split defined in Section 1.

You are interested in evaluating whether the systems give the best recommendations for each user. To accomplish this, generate the top-k (with $k = 10$) recommendation for each user in the test split. Based on the top-k recommendation list generated for each user, and using only ratings ≥ 3 in the *test* data split[6], compute:

- Hit rate at cut-off k (HR@k), averaged across users;
- Precision@k, averaged across users;
- Mean Reciprocal Rank (MRR@k);
- Catalogue Coverage;
- Discuss the advantages and disadvantages of these metrics.

Furthermore, you should:

[6]You need to convert 5 point ratings in the test split to binary labels, i.e., ratings ≤ 2 are mapped to 0 and ratings ≥ 3 are mapped to 1. For example, with Hit Rate (HR@k), if a user gave a rating ≥ 3 to one of the top-k items we recommended (i.e., the business from the test split is amongst our recommendations), then we consider that as a hit.

- Compare the two types of CF recommender systems implemented in Section 3.
- Compare the CF models from Section 3 with the CB model from Section 5.
- Compare the hybrid model from Section 6 with the other models.
- Compare also the models with respect to the baseline models. Which one works best? Why? What are the advantages and limitations of each family of approaches?

8 Exercise 8: Explanations for Recommender Systems

1. Choose 3 users for which you want to explain the top-1 recommended item by the neighborhood-based model. Create a plot to be presented for each user that illustrates how many of that user's neighbors have rated and not rated the item, respectively.
2. Perform an ablation study on the content-based recommender designed in Section 5. Report the performance (e.g., Precision@10 and HR@10) when removing one feature at a time, so you can evaluate the importance of each feature. You can base the study on the features `name` and `description`, for the same 3 users from the previous step.

9 What should be included in the report of the project

The following considerations are applicable to all sections of this project. You need to keep them in mind when you write your report.

You should include a table where you compare all the implemented recommender systems. The table rows should represent models and the table columns the evaluation measures, calculated on ratings ≥ 3 on *test.tsv*. It is not enough to report the evaluation scores of your methods. You need to **explain** the reasons why you think we see these scores. You need to show in the report that you have tried to understand why you got these specific evaluation scores, regardless of whether they are high or low.

In all cases, you should describe *what* you tried, what worked, what did not work, and *why* you think it did or did not work. For example, did you use an off-the-shelf method? How did you adapt it for the given task? Why does this adaptation work or not? Did you do some extra preprocessing or cleaning of the dataset? Was is it useful? Why or why not?

Moreover, you need to describe the limitations of what you did. What could have led to improvements in model performance? How could you have approached automatic recommendation differently? What was particularly challenging about working with this dataset and why do you think that was? This discussion does not require further experiments, but requires you to examine your experimental results, critically think about your choices and the assumptions you made, make a hypothesis on how you can overcome some limitations

and improve your solution. Furthermore, your discussion should be based on evidence, e.g., lecture material, relevant literature.

10 Extra work

The above is a description of the minimum that we expect you to implement, but you are allowed to expand upon this to explore more complex ideas. If you decide to submit extra work, that extra work will be graded as well.

11 Academic Code of Conduct

You are welcome to discuss the project with other students, but sharing of code is not permitted. Copying code directly from other students will be treated as plagiarism. Please refer to the University's plagiarism regulations if in doubt.

If you plan to use generative AI tools like ChatGPT, you must clearly declare how AI contributed to your work, either in your report or by submitting the university's declaration form available on KUnet[7]. Proper citation is required for any AI-generated content, and you are responsible for verifying the accuracy of the information. Misuse or lack of transparency may be considered academic misconduct Please refer to the University's guidelines[8] for using ChatGPT and similar technologies if in doubt.

For questions regarding the project, please ask on the Absalon discussion forum.

[7] https://kunet.ku.dk/work-areas/teaching/digital-learning/chatgpt-and-ai/new-rules-and-principles-from-september-2025/template-for-declaration/Pages/default.aspx

[8] https://kunet.ku.dk/faculty-and-department/copenhagen-university-library/library-access/Pages/AI.aspx