Web Recommender Systems 2025: Final Project

February 5, 2025

This project[1] is composed of six parts. Each part corresponds to the material taught in the corresponding week of the course. All parts of this project are compulsory, except for those marked as [Optional]. The project should be completed **individually**.

Midterm submission: You need to submit the first part of the report (only written report, no code) on Absalon by Monday, the **3rd of March at 11:59 AM (noon)**. The format of the report should be a PDF document, no more than **4 pages** (not including references, if needed).

Final submission: You need to submit the complete report (written report, revisions, code, read-me file) on Digital Exam by Thursday, the **27th of March at 11:59 AM (noon)**. The final submission includes the following:

- PDF document with the report including revisions, no more than **7 pages** (not including references, if needed). Revisions should be marked with red color.
- `.zip` file containing the **code** to run your experiments and **documentation** (`readme` file) on how to run it.

For the report, both midterm and final submission, you must use the ACL template[2]. Detailed instructions regarding the template and format can be found in Absalon.

Note that, at the end of this course, **you will present your work to the course instructors**. Oral presentations will be held physically at DIKU.

The final grade is based on an overall assessment, not an average of each component.

1 Week 6 [Feb 3 - Feb 9]: Familiarize Yourself with the Datasets

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

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

[1] Note that small modifications in the project description may be made throughout the course.

[2] `https://2021.aclweb.org/downloads/acl-ijcnlp2021-templates.zip` or theOverleaf version

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

You will use the "Musical Instruments" dataset from the Amazon Review data 2023 (5-core subset).[3] The complete data contains $\sim 213\,000$ entries, which is computationally expensive to process. Therefore, for this project, you will use a smaller version of the dataset that has been provided. The project dataset contains 14 000 item ratings from nearly 900 users on ~ 500 items.

We will use the pre-computed training and test splits *train.tsv* and *test.tsv* with 80% and 20% of the data respectively. These files can be found in Absalon. The files are in `TSV` format (fields in each row are tab-separated), you thus need read-in tools. You can use `pandas` for data wrangling. You are free to utilize other tools if you wish.

In Weeks 9 and 10 (see Sections 4 and 5), we will use the metadata file *metadata.tsv*, which contains information about the items (musical instruments). We will primarily use the `item_id` and `description` field.

During this first week you need to:

1. Download and import the dataset splits.
2. Clean both splits from missing information (e.g., missing user ID or item ID) and duplicates (cases where the same user has rated the same item multiple times) if any.
3. Double check that all users from the test split also appear in the training set, and remove those that do not appear in training.
4. Compute user and item statistics (such as distribution of ratings per user/item) 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?
5. Compute the frequency of items that were rated highly, i.e., rating ≥ 3. Count how many times each musical instrument was highly rated. We will use this as the TopPop recommender system in Week 7. Report in a table the top 5 most highly rated items with the number of rates and average rate.

2 Week 7 [Feb 10 - Feb 16]: 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.

- Based on the frequency of the most rated items computed in Week 6, implement the TopPop recommender system, which always recommends the same top-k items sorted decreasingly by the number of "high" ratings (e.g., ≥ 3) in the training split, *train.tsv*.

[3] The complete dataset and further information about it can be found on: `https://amazon-reviews-2023.github.io/data_processing/5core.html`.

- Choose at least one neighborhood-based model and one latent factor model that uses the observed user-item ratings in the training set to predict the unobserved ratings. Report your choice of models.
- Use 5-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).
- Choose an evaluation measure that is suitable for this task and justify your motivation in using it. Report the optimal hyperparameters together with the scores of your chosen measure, averaged over the 5 folds.
- Run the models with the optimal hyperparameters to the whole training set.
- Use the final models to rank the non-rated items for each user. This ranking will be used for the evaluation part next week.

3 Week 8 [Feb 17 - Feb 23]: Evaluation of Recommender Systems

In this session, we will discuss how to evaluate a recommender system. Specifically, let us evaluate all your recommender models from Week 7 on the preprocessed test data split studied in Week 6. You need to:

- Measure the error of the system's predicted likelihood of rating for the items (Root Mean Square Error, RMSE)[4].
- Discuss the limitations of this metric.

Now, we are interested not in whether the system properly predicts the rating of these items, but rather whether the system gives the best recommendations for each user. To evaluate this, generate the top-k (with $k = 10$) recommendation for each test user. Based on the top-k recommendation list generated for each user, and using the *test* data split[5], compute:

- Hit rate, averaged across users.
- Precision@k, averaged across users
- Mean Average Precision (MAP@k)
- Mean Reciprocal Rank (MRR@k)
- Coverage

[4]Note that RMSE can not be computed for TopPop

[5]You need to convert 4 and 5 point ratings in the test split to binary labels, i.e., ratings ≤ 3 are mapped to 0 and ratings ≥ 4 are mapped to 1. For example, with Hit rate, if a user gave a rating ≥ 4 to one of the top-k items we recommended (i.e., the item from the test split is among our recommendations), then we consider that as a hit.

Discuss the advantages and disadvantages of these metrics.

Compare the two types of CF recommender systems and the TopPop system that we have defined so far. Which one works best? Why? What are the advantages and limitations of each approach?

Everything above this point should be included in the submission for the mid-term deadline.

Analysis on the effect of the long-tail:

- Sort the users by number of ratings in the train set and select the top 20% and last 20% of users.
- Compute Hit rate on the test set for the 2 groups of users and discuss the difference (or similarity) between those scores.
- Sort the items by number of ratings in the train set and select the top 20% and last 20% of items.
- Compute Coverage on the test set for the 2 groups of items and discuss the difference (or similarity) between those scores.

Error analysis for the neighborhood-based CF:

- Select two users from the training set, one with high RR and one with low RR. These are your reference users. Retrieve the 10 nearest neighbours of each reference user. Print their rating history and analyse their predictions. How much overlap is there between the rating history of the reference users and their neighbours?
- For those users or items that your model performs poorly on ($RR \leq 0.05$), discuss the potential reasons behind.

The above list is not an exhaustive list. You can think of other ways of doing error analysis, e.g., using additional evaluation measures, or discussing outliers, for instance.

4 Week 9 [Feb 24 - Mar 2]: Text Representation

In this part, we will work with the text content from the dataset and will apply NLP techniques to represent items and users in a vector space.

1. Note: for this part, you are free to use all or a subset of the metadata file, e.g., only using items that are in the train/test splits or using the first n characters of the text content. Please justify your choice.
2. Select the column `description` from the metadata file and apply appropriate preprocessing to clean up the data, for example: tokenization, transformation to lowercase, stopword removal[6], or stemming. Motivate your preprocessing choices and report the vocabulary size before and after

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

preprocessing. There are many libraries you can use, including but not limited to, `NLTK`, `spaCy` or `CoreNLP` (requires Java).

3. Represent each item in the TF-IDF vector space. You can use your preferred library for text representation, for example, `scikit-learn` or `gensim`.

4. Represent each item using pretrained word embeddings (e.g., GloVe, word-2vec).

5. Explore the similarity between items within the vector spaces by computing their cosine similarity. Compare results obtained with TF-IDF and the word embeddings. Discuss what you find. Note that you can select a subset of items to better highlight the differences between the two text representations.

6. [*Optional*] Represent each item rated by the users using the word vectors from the last layer of BERT. Here, you can directly use the vectors from the pretrained version of BERT available in `huggingface`[7]. To load the model, install the `Transformers` library and `PyTorch`[8].

5 Week 10 [Mar 3 - Mar 9]: Content-Based Recommender System

In this week, we will continue using the train and test splits defined in Week 6.

- Note: similar to the task in Week 9, for this part, you are free to only use a subset of the metadata file, e.g., only using items that are in the train/test splits or using the first n characters/words of the text content. You are also welcome to use additional metadata (e.g., by crawling the internet). Please report and justify your choice.

- Transform the `description` column of each item into a TF-IDF representation or other numerical value, e.g., token-count based, that can represent the summaries. Select at least one other factor that can be used as an item feature, for example `title`. Apply the appropriate preprocessing on the features. These choices should also be reported and justified.

- After you represent each item in a vector space, represent each user in the same vector space. This can be done by using a simple average of the items the user rated. Note that you can also try to implement better ways to build the user representation.

- Calculate the user-item rating prediction for an item by using a similarity/distance metric between the user and the item representations. A metric such as cosine similarity or Euclidean distance can be used. Motivate your choice.

- Report Precision@10, MAP@10, MRR@10, hit rate and coverage using ratings ≥ 3 in the test set. Compare the results with the models from previous weeks.

[7] https://huggingface.co/bert-base-uncased
[8] https://huggingface.co/docs/transformers/installation

6 Week 11 [Mar 10 - Mar 16] Hybrid Recommender System

During this week, you can select between one the two following options.

Option 1 Create three hybrid recommender systems by combining a collaborative filtering model with a content-based model using the following three strategies:

- *Parallel combination strategy* that re-ranks the items by combining the individual rankings from the two models with some aggregation function such as the sum, average, minimum or maximum.
- *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 a level of one model is used as input to the other model.

Use each hybrid model to rank the non-rated items for each user.

Option 2 Create two hybrid recommender systems with a *parallel combination strategy* that combine a collaborative filtering model with the following:

- A content-based model, implemented as in Week 10.
- A content-based model that uses LLM generated descriptions. The descriptions should be generated using the item titles. You can use a small LLM (e.g., Gemma 2B).

Use the two hybrid models to rank the non-rated items for each user. Compute the evaluation scores and compare the two variants (original and LLM-generated metadata). Comment on the difference in using the original vs. the LLM generated description.

Regardless of the chosen option you need to Report Precision@10, MAP@10, MRR@10, the hit rate and coverage using ratings ≥ 3 in the test set. Compare the results with the models from previous weeks.

Note that Option 2 can be computationally expensive therefore we recommend using Google Colab or Kaggle notebooks to access a GPU and accelerate inference. You can find a simple tutorial to upload files and load LLMs from HuggingFace on Colab here and for Kaggle Here[9].

You can make the Option 2 more efficient by generating descriptions only for the top-k (with a large enough k) items recommended by the collaborative filtering model, instead of all items in the dataset. Then, your hybrid approach will re-rank the top-k items.

[9]Please note that if you want to access free GPUs on Kaggle, you need to verify your telephone number on the platform.

7 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 the table columns the evaluation measures, calculated on ratings ≥ 3 on *test.data*. 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 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.

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 (including parts marked as [Optional]).

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.

You are not allowed to use ChatGPT and similar applications of Large Language Models (LLM) that can generate text and code. Please refer to the University's guidelines for using ChatGPT and similar technologies if in doubt.

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