# Web Science 2023: Final Project

February 9, 2023

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. The project will be graded as a whole. 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 **6th 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 **30th of March at 11:59 AM (noon)**. The final submission includes the following:

- PDF document with the report including revisions, no more than **6 pages** (not including references, if needed). Revisions should be marked with a different text 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 should use the ACL template[2].

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.

## 1 Week 6 [Feb 6 - Feb 12]: 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 and additional supporting files and find information about them in Absalon:

`https://absalon.ku.dk/courses/64377/files/folder/project/data`

[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`

You will use the MovieLens 100K Dataset. This is a well-known benchmark dataset containing 100 000 movie ratings (1–5) from nearly 1000 users on $\sim$ 1700 movies.

We will use the pre-computed training and test splits *train.data* and *test.data* 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 `gzip` and `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 *movies_metadata.csv*, which contains information about the items (movies). We will primarily use the `id` and `overview` field[3].

Note that the `id` in *movies_metadata.csv* is not the same as the `id` in the **.data* files. Hence, we need to use the ID mapping table (*id_map.csv*).

During this first week 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 multiple times) if any. Sort the duplicate entries in ascending order by user id, movie id, and timestamp. 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 training set, and remove those that do not appear in training.
4. The *id_map.csv* mapping table contains two fields: 1) `id_in_movielens`, which is the movie ID in the MovieLens 100K dataset; and 2) `id_in_metadata`, which is the movie ID in the *movies_metadata.csv* file. For each movie in MovieLens 100K, use the mapping table to get the corresponding `overview` column in the *movies_metadata.csv* file. Not all movies have a mapping, so after steps (2) and (3) you will need to discard movies that do not have a mapping from all splits.
5. Compute user and item statistics (such as distribution of ratings per user/item, the top 5 most popular items) for the training set *train.data* and write a discussion; does the dataset have important properties that should be taken into account or that may mislead the evaluation?
6. Compute the frequency of movies that were rated highly, i.e., rating 3-5. How many times was each movie rated highly by user? We will use this as the TopPop recommender system in Week 7.

# 2 Week 7 [Feb 13 - Feb 19]: 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]More information about the other fields can be found on `https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv`

- Based on the frequency of highly-rated movies computed in Week 6, implement the TopPop recommender system, which should recommend the top-k movies with high ratings in the training split, *train.data*.
- 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 20 - Feb 26]: 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 test data split studied in Week 6. You need to:

- Measure the error of the system's predicted ratings for the movies (Root Mean Square Error, RMSE).
- Discuss the limitations of this metric.

Now, we are interested not in whether the system properly predicts the ratings of these movies, but rather whether the system gives the best recommendations for each user. To evaluate this, generate the top-k (with $k = 5$) recommendation for each test user. Based on the top-k recommendation list generated for each user, and using only ratings $\geq 3$ in the *test* data split[4], compute:

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

[4]You need to convert 5 point ratings in the test split to binary labels, i.e., ratings $\leq 2$ are mapped to 0 and ratings $\geq 3$ are mapped to 1. For example, with Hit rate, if a user gave a rating $\geq 3$ to one of the top-k items we recommended (i.e., the movie from the test split is amongst our recommendation), then we consider that as a hit.

- Coverage

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.

**Error Analysis for the neighborhood-based CF:**

- Ordered by the value of the `timestamp`, take the first and last users from the test set as reference and retrieve the 10 nearest neighbours of each reference user. Print their rate history and analyse their predictions.
- For those users or movies 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 27 - Mar 5]: Text Representation

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

1. Select the column `overview` from the metadata file and apply the following preprocessing to clean up the data: tokenization, transform to lowercase, remove stopwords[5], stemming. 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. Represent each movie in the TF-IDF vector space. You can use your preferred library for data analysis, like, for example, `scikit-learn` or `gensim`.
3. Represent each movie using pretrained word embeddings (e.g., GloVe, word2vec).
4. Explore the similarity between movies within the vector spaces by computing their cosine similarity. Compare results obtained with TF-IDF and the word embeddings. Discuss what you find.
5. [*Optional*] Represent each movie 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`[6]. To load the model, install the `Transformers` library and `PyTorch`[7].

[5] Lists of stopwords for different languages: http://members.unine.ch/jacques.savoy/clef/
[6] https://huggingface.co/bert-base-uncased
[7] https://huggingface.co/docs/transformers/installation

# 5 Week 10 [Mar 6 - Mar 12]: Content-Based Recommender System

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

- Transform the `overview` column of each item into a TF-IDF score or other numerical value, e.g., token-count based, that can represent the summaries. Select other factors that can be used as item features, for example `genres` and `production_companies`.
- After you represent each movie in a vector space, represent each user in the same vector space. This can be done by using an average of the items the user rates. Instead of a simple average, a weighted average can be used. Note: the user representations and item representations all have the same number of dimensions.
- Calculate the user-item rating for an item by using a similarity metric between the user and the item. A similarity metric such as cosine distance or Euclidean distance can be used.
- Report Precision@5, MAP@5, MRR@5, hit rate and coverage using ratings $\geq 3$ in the test set. Compare the results with the models from previous weeks.

# 6 Week 11 [Mar 15 - Mar 19] Hybrid Recommender System

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

1. *Weighted strategy* that re-ranks the items by combining the individual rankings from the two models with some aggregate function such as the sum, average, minimum or maximum.
2. *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.
3. *Meta-level 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.

- Report Precision@5, MAP@5, MRR@5, the hit rate and coverage using ratings $\geq 3$ in the test set. Compare the results with the models from previous weeks.

# 7 What should be included in the report of the project

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 $\geq 3$ on *test.data*

You should describe what you tried, what worked, what did not work, and why you think it did or did not work, for each part of the project. 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.

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