Web Science 2022: Final Project

February 8, 2022

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**. 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 final submission is a `.zip` file, which should contain each of the items listed above. The submission deadline is no later than **Wednesday 30 March 2022 at 12h00 (noon)**. The format of the report should be a PDF document using the ACL template[2], no more than **6 pages** (not including references, if needed).

Note that, at the end of this course, **you will present your work to the course instructors**.

1 Week 6 [Feb 7 - Feb 13]: 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 will use Amazon Review Data (2018). You can download the dataset and find information about it here:

`https://nijianmo.github.io/amazon/index.html`

The complete dataset contains millions of entries, which can be computationally expensive to process. We will use one of the "Small" (5-core) subsets, specifically the "Software" category. A reference to this dataset subset (and other dataset details) can be found on the linked website page.

In weeks 9 and 10 (see Sections 4 and 5), we will use the metadata file for the "Software" category, which can be downloaded from the section "Complete review data", at the same link provided above. You need to fill in a form with

[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`

some data to download the metadata dataset. This is needed to keep track of the dataset users and guarantee that the dataset is not used for commercial purposes. You can use your KU email and specify “University of Copenhagen” as affiliation.

The dataset is in `JSON` format, you thus need read-in tools. You can use the built-in Google Colab notebook to pre-process the dataset or subsets through this link: `https://colab.research.google.com/drive/1Zv6MARGQcrBbLHyjPVVMZVnRWsRnVMpV`. The notebook uses gzip and pandas for data wrangling. You are free to utilize other tools if you wish.

During this first week you need to:

1. Download and import the dataset.
2. Clean the dataset from missing ratings and duplicates (cases where the same user has rated the same item multiple times) if any.
3. Later in the project, we will evaluate different recommender systems on the task of recommending the next item a user will like. For that purpose, create a test set by extracting the latest positively rated item (rating ≥ 4) by each user. Remove users that do not appear in the training set.
4. Compute user and item statistics (such as distribution of ratings per user/item, the top 5 most popular items) for the training set and write a discussion; does the dataset have important properties that should be taken into account or that may mislead the evaluation?
5. The metadata file contains information of all items in the complete dataset, not only the 5-core subset. You need to filter out all items that are not included in the training and/or test sets after steps (2) and (3).

2 Week 7 [Feb 14 - Feb 20]: 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/`.

- Define a neighborhood-based and a 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 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 optimal hyperparameters together with the corresponding validation Root Mean Square Errors averaged over the 3 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 21 - Feb 27]: Evaluation of Recommender Systems

In this session, we will discuss how to evaluate a recommender system. Specifically, let us evaluate your Collaborative Filtering (CF) models on the *test* data split defined in Week 6. You need to:

- Measure the error of the system's predicted ratings for Software products (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 products, 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 and compute:

- Precision@k, averaged across users
- Mean Average Precision (MAP@k)
- Mean Reciprocal Rank (MRR@k)
- Discuss the advantages and disadvantages of these metrics.

Based on the top-k recommendation list generated for each user, compute the averaged hit rate of the model. Remember that, if a user rated one of the top-k we recommended (i.e., the product from the test split is amongst our recommendation), then we consider that as a hit.

- Compute the system's hit rate averaged over the total number of users in the test set.
- Compare the two types of CF recommender systems that we've defined so far. Which one works best? Why? What are the advantages and limitations of each approach?

Error Analysis for the neighborhood-based CF:

- Ordered by the value of the column "unixReviewTime", 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 products 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 28 - Mar 6]: Text Representation

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

1. Select the column "title" from the metadata (only for the products rated in our small subset of the dataset) and apply the following preprocessing to clean up the data: tokenization, transform to lowercase, remove stopwords[3], 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 product in the TF-IDF vector space. You can use your preferred library for data analysis, like, for example, sikit-learn or gensim.

3. Represent each product using pretrained word embeddings (e.g., GloVe, word2vec).

4. Explore the similarity between summaries for the same product by computing their cosine similarity, and compare results obtained with TF-IDF and the word embeddings. Discuss what you find.

5. [*Optional*] Represent each product 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[4]. To load the model, install the Transformers library and PyTorch[5].

5 Week 10 [Mar 7 - Mar 13]: Content-Based Recommender System

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

- Transform the "title" column of each product 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 product features.

- After you represent each product 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, where the weight is the rating for the item by the user. 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@k and the hit rate using the test set. Compare the results with the models from previous weeks.

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

6 Week 11 [Mar 14 - Mar 20] 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 and the hit rate using 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.

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.