Recommender Systems SS2022: Group 1

DACHS FABIAN, 01627961
DENZEL JONATHAN, 12127086
MATHIS TOBIAS GALLUS, 01621473
PAJKOVSKI MARKO, 12126329
PLETIKOSIC VICE, 51831449

Recommender Systems play an important role for online shopping websites and can lead to increased revenue. In this paper several models for a recommender system are compared, combined and evaluated to recommend items for a fashion website according to the 2022 Dressipi-Challenge.

CCS Concepts: • **Information systems** → **Recommender systems**; **Collaborative filtering**.

Additional Key Words and Phrases: datasets, recsys, dressipi, collaborative filtering, recommender systems

1 INTRODUCTION

The fashion industry is a very competitive industry and to gain a competitive advantage the use of recommender systems can increase the revenue of e-Commerce platforms. One of the challenges for recommender systems on those platforms is the lack of personal data for customers, that are not logged in or are new to the website. The data that the website collects within a single session is sometimes everything that is available to build a recommender system model. The AI company Dressipi posted over 1.1 million sessions from over 18 months in a competition to build a recommender system for their online services which they sell to other companies. Dressipi sells analytic services to clothes and footwear stores to increase revenue and reduce returns. While this project focuses on forecasting sales, Dressipi also offers automated feature tagging, automated intentional tagging and automated personalised style recommendations.

This project "is based on the ACM RecSys 2022 Challenge, using real data from Dressipi." -Tuwel RecSys

"Dressipi is the fashion industry's artificial intelligence expert, providing product and outfit recommendations to leading global retailers. " - RecSys Challenge 2022

The goal of the challenge is to predict accurately which fashion item will be bought at the end of the session to show those recommendations to a user and lead them to buy (more) items. This might lead to more successful conversions, more products sold and a higher customer satisfaction.

2 DATA (DESCRIPTION OF THE DATA)

The data is accessible on the website: www.dressipi-recsys2022.com. It consists of the following four data types split onto several CSV-files.

2.1 Data Types

Sessions. `session_id, item_id, date`

All sessions consist of a unique identifier and timestamped item identifier. There were several session datasets: for training, the challenge leaderbord and for the final evaluation of the challenge. The latter two were not as relevant for the academic task of this course, but were used to evaluate the models online

Authors' addresses: Dachs Fabian, 01627961; Denzel Jonathan, 12127086; Mathis Tobias Gallus, 01621473; Pajkovski Marko, 12126329; Pletikosic Vice, 51831449.

Purchases. `session_id, item_id`

The purchases consist of the session identifier to which the purchase belongs to and the item identifier, that represents the bought item.

Candidate Items. `item_id`

The candidate items are just a subset of item identifier that are items to be recommended to limit the amount of possible recommendations. They include roughly 17% of all item identifier, limiting the possible recommendations to one sixth of all items.

Item Features. `item_id, feature_category_id, feature_value_id`

The item feature data consisted of a item identifier, a feature identifier and a feature value identifier. No descriptive information about any actual value is included, limiting the room for interpretation of any value. Most likely these features consist of values such as category: dress; color: blue; etc.

2.2 Data structure

The provided data was free from missing values. However, we preprocessed the data to have it in the form of a dictionary with the following structure: `{session_id:[item_list]}`. There is no explicit feedback available, all information must be extracted from implicit feedback. The timestamp-column was not further used in our considerations.

2.3 Data exploration

The following paragraphs contain questions that were answered during the exploration.

How much is the overlap of purchases if the session is similar? We analysed the test sessions for their closest neighbor and tried to find similar session. The results were meager. Building similarity for session did not have a big overlap of purchases.

What and how long is the longest session? From table 1 we take that there are huge differences between sessions, having a range from only 1 item to 100 items. Most of the time the sessions consist of fewer than 10 items. The longest session in the dataset includes 100 items and has the ID 54433. The distribution of the length of sessions can be seen in the figure 1.

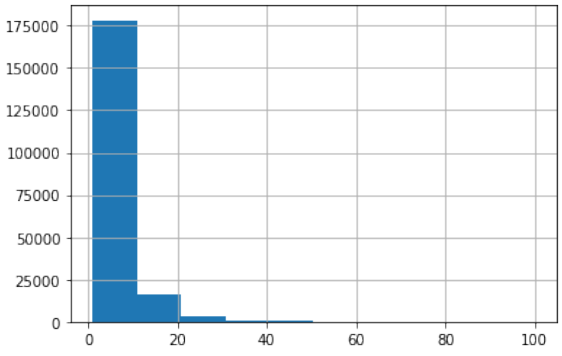

Fig. 1. The length of sessions on the X-Axis and the count of each session-length on the Y-Axis

number of sessions	min session length	max session length	mean session length	std session length
199341	1	100	4.76	6.13

Table 1. Metadata of sessions.csv

How unequal are purchases distributed amongst items? An analysis of the purchases data shows that there are only a few items that are bought very frequently. For example the most popular item was bought more than 8000 times which makes up around 0.8% of all purchases. This unbalance yields in the effect that 33% of items explain 80% of sessions.

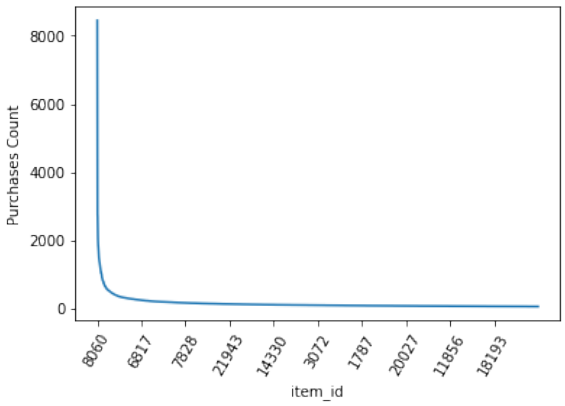

Fig. 2. The distribution of purchases of the top 4990 items.

3 APPROACH

In the first iteration we specified our technology stack to be based on python3, pandas and GitLab as used in the lectures. Furthermore our internal organisation was managed in a Google Doc, document. Our communication is running on a WhatsApp group and Zoom/Google-Meet conferences.

3.1 Implementation

As a first step we designed a standard `pipeline` in order to ensure that the structure of the project remains consistent across the different models and methods. The standard pipeline consists of a parent class called `BasisRecommender`, which every other recommender has to inherit from. This ensures that every recommender has at least 2 functions:

- `train(X_train,y_train)`: used to fit the recommender
- `predict(X_test)`: used to make the recommendations

All our experiments consists of two parts: First, we test the specific recommender locally on our train-test split using the `train_test_split()` function, which splits the training data with a ratio 0.95 at random. In the second part we use `train_test_split_challenge()` which uses the full training set and the data in `test_leaderboard_sessions.csv` for the test data. The online evaluation is done via upload to the dressipi-recsys2022 challenge website, the local evaluation is computed locally.

3.2 Performance Measurements

To evaluate our recommenders locally we implemented an evaluation script. This script uses the test set to calculate the following performance indicators:

- **MRR**: Mean Reciprocal Rank calculates as follows, where Q is the number of recommendations:

$$\text{MRR} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{\text{rank}_i}$$

- $\textbf{mean}_{pos}$: The mean position of the true item in the ranking
- $\textbf{std}_{pos}$: The mean position of the true item in the ranking
- $\textbf{accuracy}_{all}$: The accuracy of whether the item is in the recommended items
- $\textbf{accuracy}_{100}$: The accuracy of whether the item is in top 100 the recommended items
- $\textbf{accuracy}_{50}$: The accuracy of whether the item is in top 50 the recommended items
- $\textbf{accuracy}_{10}$: The accuracy of whether the item is in top 10 the recommended items

4 DESCRIPTION METHODS USED

4.1 Baseline Recommender

The logic behind the Baseline Recommender is straight-forward and rather simple: The Recommender only looks at the purchased items and order them in a way so that the most purchased items are at the top, takes the top 100 items and then recommend them for all sessions. This approach is model-agnostic and takes no further session-information into account. However, it is well suited as a baseline. The results can be taken from table 2.

4.2 Content-Based Recommender

We also implemented a recommender that makes use of the item features. During training we first find the 2000 most bought items and store the features of those items during training. Further on we will only use those as the candidate items.

When we want to recommend items from a session information, we try to rank the candidate items by how much the features of the items in the session overlap with the features of the candidate item. This overlap error will be discussed below. These candidate items are then sorted by the overlap and the top 100 features will be recommended.

As an error function between the session features and the features of the candidate item we came up with the following function, where N is the number of items in the session, F is the number of features and $x_{i,f}$ equals 1 if the feature is in the candidate item and 0 otherwise:

$$\text{overlap error} = \sum_{i=0}^{N}\sum_{f=0}^{F} -10 \cdot x_{i,f} + 1 \cdot (1 - x_{i,f})$$

As this method has to calculate for every recommendation the feature error for every candidate item, this method can be quite slow for large queries of recommendations.

The performance of this model on the different performance indicators are listed in table 2.

4.3 Session Similarity Recommender

The session similarity recommender analyses the items within a session. Items that appear often together in a session are then recommended. This recommender analyses the whole session and treats all items equally including the purchases, by simply adding the purchases into the sessions dictionary. Then it counts how often items appear together in a session across all sessions in an item-item matrix. This is done during the training step.

In the recommending method for every item within a test session the items that were bought with this item are added to an item vector. This vector then has many entries where each entry represents the sum of how often any item from a test session was together in a session with one candidate item.

A high value in this vector means that these items were often in the same session with all the current items in the recommended session.

The 100 items with the highest values in this item-vector are then recommended in descending order.

4.4 Item-Item Model

The item-item recommender was developed independently from the other models, but adds on the session similarity recommender, by using the core logic of counting the items that occur in a session together.The main difference is that it includes the classic mathematical metric: cosine similarity, which takes into consideration items which have not been bought so often but are highly similar to other items. During the training step it computes the similarity matrix for each item, which is then used to recommend the top 100 closest items.

$$Similarity_{IJ} = \frac{\sum_{i=1}^{n} I_i \cdot J_i}{\sqrt{\sum_{i=1}^{n} I_i^2} \cdot \sqrt{\sum_{i=1}^{n} J_i^2}}$$

4.5 Hybrid Recommender

The Hybrid Recommender Model is an ensemble of different models. As input it takes a list of dataframes which are the predicted results of previous models and predicts a new model.

Generally, it combines the data frames into one and then creates a new rank by dividing 1 and the previous MRR rank. It then uses the top 100 as the new prediction.

Here one could refine this model by adding weighting options to lists. Or one could use a nearest neighbor approach for the models.

For this project we tested two hybrid versions. One using the Baseline Recommender, Content-Based Recommender and Session Similarity Recommender as Hybrid1. The other using only Content-Based Recommender and Session Similarity Recommender as Hybrid2.

5 RESULTS

The results from our models can be read from the table 2. The performance of our models is quite different, our best model can achieve an MRR of 0.168 on the official RecSys 2022 leaderboard which can compete with the #50 recommenders submitted in the RecSys challenge. Our Session-Simmilarity recommender can achieve an MRR of 0.0976, the Contend-Based recommender an MRR of 0.056. Our hybrid models that represent different ensembles of the models can achieve an MRR of 0.0832 and 0.0932 on the official RecSys 2022 leaderboard.

Method	MRR (RSC)	MRR (local)	mean_{pos}	std_{pos}	accuracy_{100}	accuracy_{50}	accuracy_{10}
Baseline	0.016	0.017	35.610	29.209	0.122	0.086	0.031
Content-based	0.056	0.045	26.901	27.235	0.262	0.208	0.109
Session Simmilarity	0.0976	0.0691	19.921	24.136	0.241	0.209	0.131
Item-Item	0.168	0.073	14.654	19.059	0.232	0.215	0.143
Hybrid 1	0.0832	0.06310	25.6843	26.7665	0.32072	0.26033	0.1351
Hybrid 2	0.0932	0.0699	24.2204	26.2603	0.3276	0.2697	0.1513

Table 2. Results of all methods

6 CONCLUSION

All in all, our results were satisfying. However, comparing the results with the results of the winner of the challenge we recognise that there is still room for improvement.

We therefore want to give here some ideas, what we think might improve the performance. To begin with, it might certainly be useful to look at some unique characteristics of the fashion domain. The following bullet points lists some characteristics and states to what extent it could improve our models:

- seasonality of purchases: By taking into account in which season the session takes place, this would probably yield better results. It is certainly more likely that winter coats are bought in the cold seasons than in the summer.
- user-aware: By knowing the sex, age, location of the customer, recommendation could be based on those demographics (e.g: dresses vs. suits).
- trends: items which were popular some time ago, are not necessarily still popular today. A possible solution would be to put more weight to the recent history
- complementary recommendation: this is not really feasible with this data, because we only have access to one session and there is no information of items in some shopping cart, but for the fashion domain in general it is certainly useful to look at complementary items (e.g: trousers with matching shoes)

All our models perform quite differently, with MRR going from 0.016 up to 0.168, but for all of the models we can see that the mean position is around 20 with a standard deviation of around 25. Affect that the position, if present in the ranking list is in most cases lower than 50, can also be observed, as the accuracy of the top 100 ranked items and the accuracy of the top 50 items only decreases slightly. Therefore we suggested that a hybrid model could be very successful. As the performance of the hybrid model does not increase our overall performance by much, we suggest there is a large overlap in the items recommended and the recommendations are not very different from another, even tough the precision in some models is better than in others.

Within another extension of our recommender the hybrid model could be based on a neural network as described in the lecture to determine which recommender or which combination of recommenders is best suited for the given instance. This might improve the blend of recommenders that is described in the section 4.5.

7 DISTRIBUTION OF WORK

The workload was fairly distributed amongst the team members in the following ways. Whilst selected individuals were responsible for certain tasks, during all steps we had meetings together and worked as a team.

7.1 Data exploration

Everyone in our group did their own data exploration and we discussed each others results together. We used different approaches with some out-of-the-box data exploration libraries (**sweetviz**) and going through the different csv files to explore relations within the data.

7.2 Recommender modeling

We discussed several recommender designs together and ended up with the described models. Tobias worked on the Baseline recommender and the pipeline of all recommenders. Fabian worked on the Content-based recommender. Jonathan worked on the Session-Similarity recommender. Marko worked on the Item-Item recommender. Vice worked on the hybrid recommender. Fabian implemented the evaluation of the models.

7.3 Format and submission

Fabian, Vice and Tobias worked on the final submission format.

7.4 Report

The coordination, content creation and fine tuning of the report was the responsibility of Marko and Jonathan.