Collaborative Filtering: Matrix factorization

Pr. El Habib Nfaoui

Collaborative Filtering: Matrix Factorization

- In general, the idea of **matrix factorization** is to decompose a matrix ***M*** into the product of several factor matrices, i.e.,

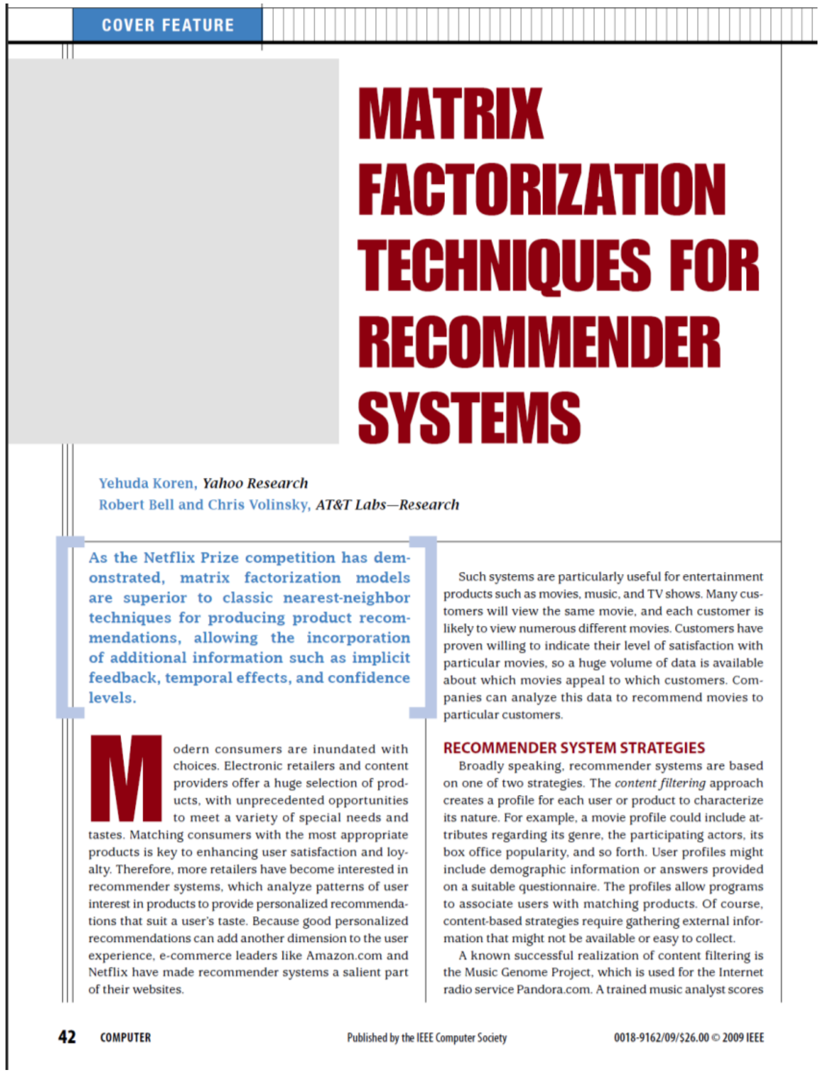
  $$\boldsymbol{M} = \boldsymbol{F}_1 \boldsymbol{F}_2 \cdots \boldsymbol{F}_n$$

  where *n* can be any number, but it is usually 2 or 3.
- Part of its success is due to the Netflix Prize contest (2009) for movie recommendation, which popularized a **Singular Value Decomposition (SVD)** based matrix factorization algorithm. Since then various matrix factorization methods have been studied for collaborative filtering, including Nonnegative Matrix Factorization, Maximum Margin Matrix Factorization, and Probabilistic Matrix Factorization. The prize winning method of the Netflix Prize contest employed an adapted version of SVD called timeSVD++.

- Netflix Movie Dataset
  - 100 million ratings that 500,000 users gave to 17,000 movies

- Grand Prize of 1 Million $ won by team from Yahoo and AT&T
  - beating Netflix's neighborhood based method by 10% (**root mean square error** (RMSE) performance)
  - using matrix factorization extended with modelling of biases and temporal dynamics

Collaborative Filtering: Matrix Factorization



COVER FEATURE

MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS

Yehuda Koren, *Yahoo Research*
Robert Bell and Chris Volinsky, *AT&T Labs—Research*

As the Netflix Prize competition has demonstrated, matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels.

Modern consumers are inundated with choices. Electronic retailers and content providers offer a huge selection of products, with unprecedented opportunities to meet a variety of special needs and tastes. Matching consumers with the most appropriate products is key to enhancing user satisfaction and loyalty. Therefore, more retailers have become interested in recommender systems, which analyze patterns of user interest in products to provide personalized recommendations that suit a user's taste. Because good personalized recommendations can add another dimension to the user experience, e-commerce leaders like Amazon.com and Netflix have made recommender systems a salient part of their websites.

Such systems are particularly useful for entertainment products such as movies, music, and TV shows. Many customers will view the same movie, and each customer is likely to view numerous different movies. Customers have proven willing to indicate their level of satisfaction with particular movies, so a huge volume of data is available about which movies appeal to which customers. Companies can analyze this data to recommend movies to particular customers.

RECOMMENDER SYSTEM STRATEGIES

Broadly speaking, recommender systems are based on one of two strategies. The *content filtering* approach creates a profile for each user or product to characterize its nature. For example, a movie profile could include attributes regarding its genre, the participating actors, its box office popularity, and so forth. User profiles might include demographic information or answers provided on a suitable questionnaire. The profiles allow programs to associate users with matching products. Of course, content-based strategies require gathering external information that might not be available or easy to collect.

A known successful realization of content filtering is the Music Genome Project, which is used for the Internet radio service Pandora.com. A trained music analyst scores

Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

Latent factor models

- For collaborative filtering, matrix factorization belongs to a class of latent factor models. In such models, **latent variables** (also called **features, aspects, or factors**) are introduced to account for the underlying reasons of a user purchasing/using a product. **When the connections between the latent variables and observed variables (user, product, rating, etc.) are estimated during the training phase, recommendations can be made to users by computing their possible interactions with each product through the latent variables.**
- Item characteristics and user preferences are represented as numerical factor values in the same space
  - some latent factors are human understandable, others are not
  - The number of **latent factors f** is set as hyperparameter
  - **40 to 1500 factors were used by Netflix Challenge winners**
- Ratings $\hat{r}_{ui}$ are estimated as the dot product of the user and item factor values

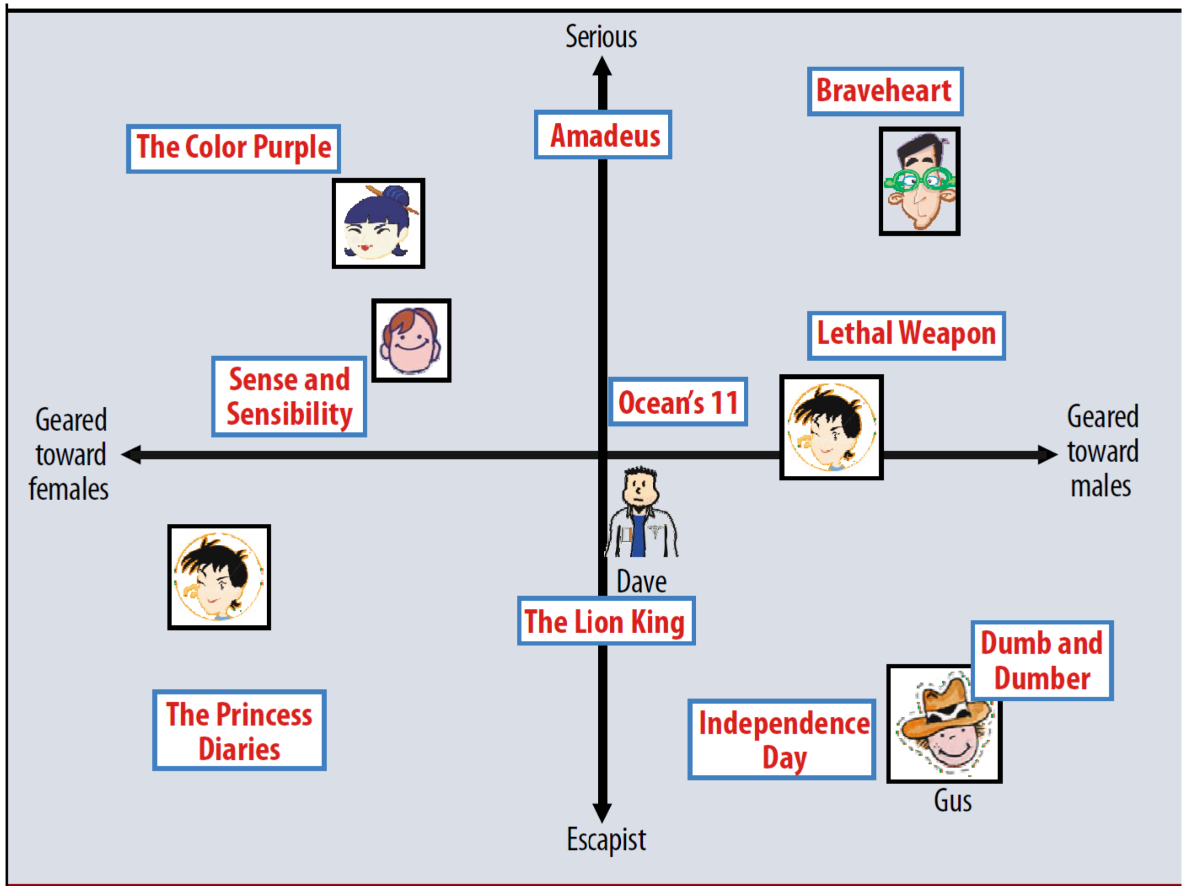


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

Matrix Factorization

- Matrix factorization is based on **the assumption that the interests of users can be described in a low-dimensional space of latent factors** that synthesize the subjacent properties of items that determine why a person may like them [Koren et al. 2009].

- In this perspective, **users and items are assumed** to be describable as vectors in a common **latent factor vector space**, in such a way that users with similar tastes would have similar factor vectors, and so would items that please similar users.

- These **low-dimensional vectors** have come to be **referred** to as **embeddings** in the recent literature, making a connection to similar techniques in fields such as natural language processing and information retrieval [Mikolov et al. 2013, Pennington et al. 2014].

Embeddings:

Embeddings are **low-dimensional (dense)**, **learned representations** of objects in the form of vectors of real numbers.

Matrix Factorization

- The **latent factors do not necessarily correspond to actual properties of items or user traits** as we would probably represent them in our human understanding. They are handled as **abstract dimensions** for which users and items are given values by an algorithmic approach, and are usually **not interpretable by eye inspection or a clear natural intuition**.

- Matrix factorization algorithms only require **deciding how many factors we wish consider**—which becomes a **hyperparameter** of the approach—but not what these factors are, other than axes of a multidimensional vector space.

Matrix Factorization: Formulation

- Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space.
  - each item *i* is associated with a $q_i \in \mathbb{R}^f$, and each user *u* is associated with a vector $p_u \in \mathbb{R}^f$

    In this case, we assume the presence of latent factors for both the users and the items. In other words, each item will be described by only f aspects (factors) values (i.e. new features) indicating how much values indicating how much that item exemplifies each aspect. Correspondingly, each user is also described by ***f aspect values*** indicating how much he/she prefers each aspect.
  - For a given item i, elements of q_i measure the extent to which the item possesses those factors, positive or negative.
  - For a given user u, elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative.

$p_u = (p_{u1}, p_{u2}, \ldots\ldots, p_{uf})$ where $p_{uj} \in \mathbb{R}$

$q_i = (q_{i1}, q_{i2}, \ldots\ldots, q_{if})$ where $q_{ij} \in \mathbb{R}$

We don't know the value of each vector component (for this reason they are called latent). The system have to learn the factor vectors (p_u and q_i).

- User-item interactions are modelled as **dot product** in that space

The resulting dot product, $q_i^T p_u$ captures the interaction between user u and item i—the user's overall interest in the item's characteristics. This approximates user u's rating of item i, which is denoted by r_{ui}, leading to the estimate

$$\hat{r}_{ui} = q_i^T p_u$$

Matrix Factorization: Formulation

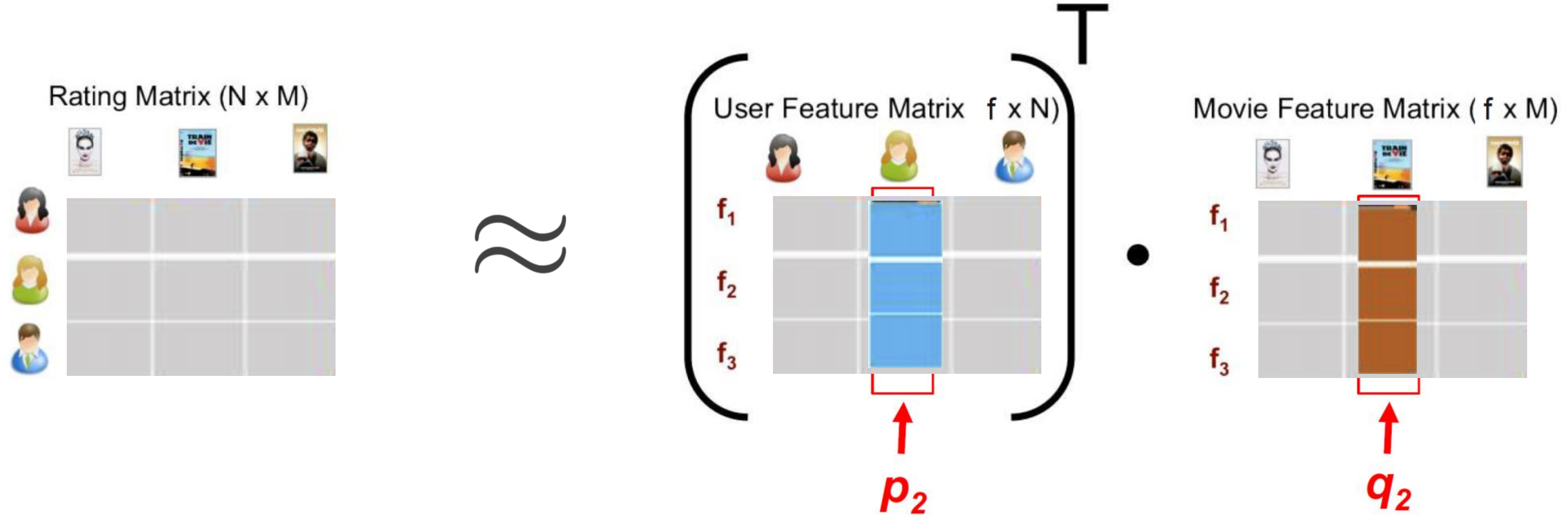
- The major challenge is computing the mapping of each item and user to factor vectors $q_i, p_u \in \mathbb{R}^f$
  After the recommender system completes this mapping, it can easily estimate the rating a user will give to any item by using Equation above.

- Such a model is closely related to singular value decomposition (SVD).
- Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix. This often raises difficulties due to the high portion of missing values caused by sparseness in the user-item ratings matrix. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting.

Matrix Factorization: Formulation

⇒ How to learn the mapping of items and users to the corresponding factor vectors ?

- **Approach**: approximately **decompose rating matrix** into dot product of user feature and item feature matrices



- Rating matrix is usually sparse

The objective function

- To learn the factor vectors (p_u and q_i), a solution is to minimize the **squared error** on the **set of known ratings**

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2$$

Here, κ is the set of the (u,i) pairs for which r_{ui} is known (the training set).

r_{ui} is the known rating of user u for item *i*

$\hat{r}_{ui} = q_i^T p_u$ is the predicted rating

- We add a regularization term to avoid overfitting

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

λ (Lambda) is a hyperparameter to control the extend of the regularization

Learning algorithms

- Two approaches to minimizing the previous equation are *stochastic gradient descent* and *alternating least squares* (ALS). We will focus on the *stochastic gradient descent*.
- Simon Funk popularized a stochastic gradient descent optimization of the previous equation* :
  - the algorithm loops through all ratings in the training set. For each given training case, the system predicts r_{ui} and computes the associated prediction error:

$$e_{ui} \overset{def}{=} r_{ui} - q_i^T p_u$$

  - Then it modifies the parameters by a magnitude proportional to momentum Gamma $\boldsymbol{\gamma}$ in the opposite direction of the gradient, yielding:

- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$
- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$

* http://sifter.org/~simon/journal/20061211.html

Adding Biases: Item and User Bias

- The previous equation tries to capture the interactions between users and items that produce the different rating values. However, much of the observed variation in rating values is due to effects associated with either users or items, known as **biases** or intercepts, independent of any interactions.
  - For example:
    - Typical collaborative filtering data exhibits large systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others.
    - Some products are widely perceived as better (or worse) than others.
- Explicitly modelling the biases improves the model performance:
  - A first-order approximation of the bias involved in rating r_{ui} is as follows:

$$b_{ui} = \mu + b_i + b_u$$

μ is the overall average rating

b_i and b_u indicate the observed deviations of user *u* and item *i* from the overall average

Example: Item and User Bias

$$b_{ui} = \mu + b_i + b_u$$

- The average rating over all movies, μ, is 3.7 stars
- *Titanic* is better than an average movie, so it tends to be rated 0.5 stars above the average (b_i)
- Joe is a critical user, who tends to rate 0.3 stars lower than the average (b_u).
- The bias for *Titanic*'s rating by Joe would be 3.9 stars (3.7 + 0.5 - 0.3)

Adding Item and User Biases to the Model

To include biases, the equations for predicting ratings and learning latent factor vectors are extended as follows

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u \longleftarrow$$ **Biases + Item/user interaction**

$$\min_{p*,q*,b*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda$$

$$(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

Matrix factorization models' accuracy on the Netflix Challenge

- Winning team further extended the model with
  - implicit feedback in addition to ratings to overcome cold start problem
  - temporal dynamics: change of user preferences and biases over time
- Results show that matrix factorization techniques
  - outperform KNN
  - scale to large use cases
  - allow flexible modelling of use case

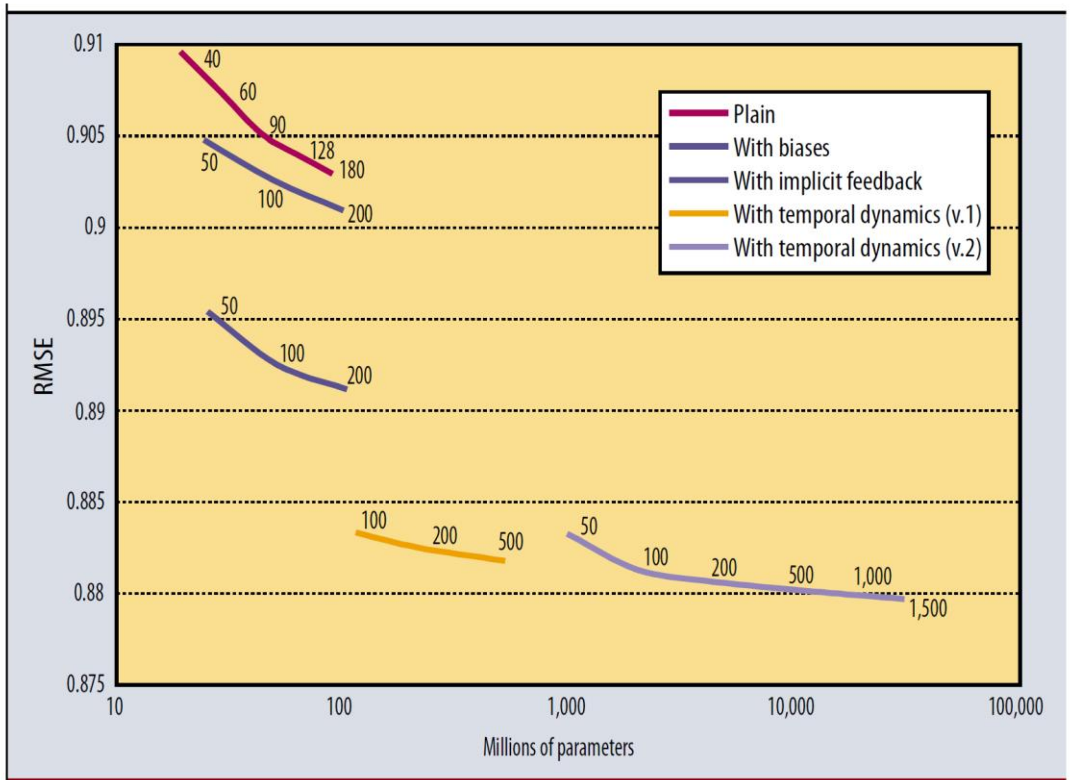


Figure 4. Matrix factorization models' accuracy. The plots show the root-mean-square error of each of four individual factor models (lower is better). Accuracy improves when the factor model's dimensionality (denoted by numbers on the charts) increases. In addition, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves RMSE = 0.9514 on the same dataset, while the grand prize's required accuracy is RMSE = 0.8563.

References

- Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.
- Castells, P., & Jannach, D. (2023). Recommender Systems: A Primer. ArXiv. https://arxiv.org/abs/2302.02579
- C. Bizer. (2023). Recommender Systems – Part 2. Course Slides.