How to optimize product recommendations using machine learning

Xiaomin Jin

Paper for Scientific Writing Class II

Department of Computer Science
Darmstadt University of Applied Science
Germany
Summer term 2022

Lecturers: Prof. Bettina Harriehausen-Muehlbauer, Prof. Katja Lenz

Contents

Acronyms

AI artificial intelligence. 5

CBF content-based filtering. 7

CF collaborative filtering. 7

HBF hybrid-based filtering. 7

ML machine learning. 5

MTF matrix factorization. 12

PRE product recommendation engines. 5

RE recommendation engines. 5

SVD single value decomposition. 12

Abstract

In the last two decades, e-commerce has developed itself from a niche market to one of the most prominent players in product distribution. Nowadays, online shops offer an enormous amount of products to their customers. With a large amount of data, it has become a challenge to process and provide valuable data to customers to fulfill their needs and choice of products. Big players like Amazon have been using recommendation engines with machine learning algorithms like collaborative filtering, hybrid filtering, content-based filtering, etc., to overcome real-time product recommendations for their customers. This paper will look at state-of-the-art recommendation engines and their algorithms. We will determine how to optimize a recommendation engine which recommends consumers quality products.

1 Introduction

1.1 What is a recommendation engine?

Recommendation engines (RE) in general, also known as recommendation systems or recommendation agents, are software engines that use algorithms to give customers a tailored experience. RE are used in a variety of areas like e-commerce, social media, streaming services (e.g., Netflix, Amazon Prime, Spotify), and education, where RE provide advice to students on which universities are most suitable for them to attend according to their interests and skills [1], knowledge management inside an organization such as suggesting solutions to a known problem [2]. When it comes to the area of e-commerce, there are two different recommendation types, product recommendation and vendor recommendation [3].

Product recommendation engines (PRE) use machine learning (ML) and artificial intelligence (AI) to recommend to customers products that they will most likely buy depending on their shopping behavior. These behaviors include purchase history, search behavior, product preferences, wish lists, geographic location, age, and gender. In addition, the engine can consider the feedback on most-viewed items, trending items, and feedback of other users, too [4]. In this paper, we will look deeper into the current technologies of PRE by understanding the different algorithms.

1.2 Motivation

E-commerce has been around for more than 40 years now. It started in the 1960s with the founding of CompuServe in 1969 but kicked off in 2005 after the founding of Amazon and eBay in 1995. Both Amazon and eBay had annual revenue of around 15 billion U.S. dollars (Figure 1) together. Since then, online shopping has severely affected our shopping behavior and how we consume products and services surveyed in [5] and [6].

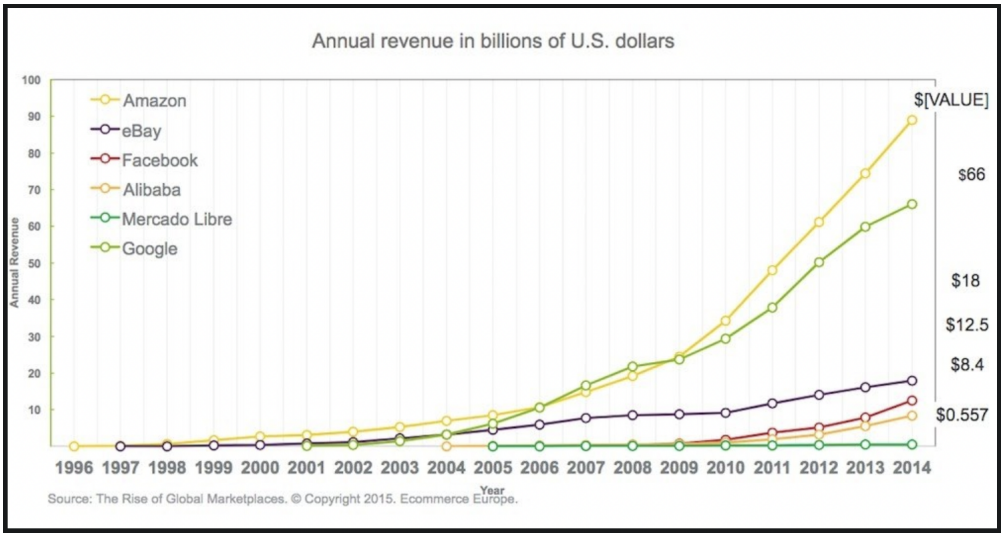


Figure 1: Annual revenue in billions of U.S. dollar from 1996-2014 [7], see appendix eComRevenue1996to2014.png on microSD

According to BigCommerce [5] around 1.8 billion people worldwide bought items from online markets in 2018. United Nations Conference on Trade and Development (UNCTAD) Secretary-General Mukhisa Kituyi [8] stated that "the COVID-19 pandemic accelerated the shift towards a more digital world. The changes we make now will have lasting effects as the world economy begins to recover [...]". Surveys made by Statista [9], UNCTAD, and Netcomm Suisse (eCommerce Association) [10] mentioned that due to the pandemic, people now visit online shops more often and rely much more on online shops than ever before. Since the pandemic, consumers do not only order classic products like flight tickets or dinner over delivery services. There has been an increase of six to ten percent across most consumer product categories (Figure 2).

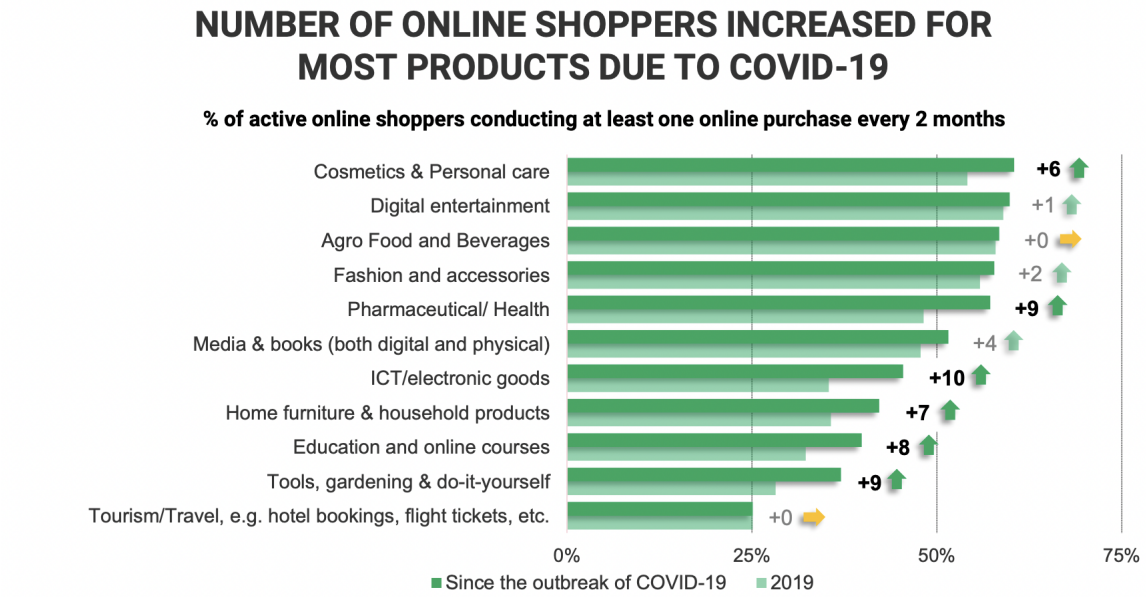


Figure 2: Increase of consumers buying products since 2019 [10], see appendix eCom-COVID19UNCTAD.png on microSD

German food e-commerce [11] registered an increase in customers buying food online instead of going to groceries. The revenue has increased to 772 million euros in the second quarter of 2020, which is 91% higher than in 2019. A variety of new food delivery companies were founded during the pandemic. Companies like Flink offer products you can buy in grocery stores as a delivery service.

With the rise of consumers, companies will want to offer more products online to satisfy the needs of consumers. Amazon offers more than 12 million products on its own, and if we count in Amazon Marketplace products, the amount would be at 350 million products [5]. However, how do customers know which of these products are most suited for them?

Big online players like Amazon, Netflix, Google, etc., have been using PRE for over a decade. The system is not perfect. Amazon, e.g., shows a variety of products concerning this specific product. Often these recommended products vary in quality. Customers are offered products with mixed reviews from zero to five stars in Figure

3. Torres-Moraga [12] connected that a customer's loyalty depends on the quality of a product.

Figure 3: Recommended products with mixed reviews, see appendix amazon-PREmixedReview.png on microSD

These lead to whether an optimized PRE can lead to better consumer satisfaction and higher revenue. This paper will explore the basics of a PRE and look at the current state-of-the-art recommendation algorithms (RA). Secondly, we will see how to implement such a system with an optimized RA, and we will present our results and conclusions.

2 State-Of-The-Art Research

2.1 The Fundamentals of product recommendation engines

This chapter will look into the currently available product recommendation engines. Geetha [13] mentions that "[...] users and items are the two major topmost objects that play a vital role." User feedback (likes, dislikes), purchases, and searches serve as input information. These informations are stored in a rating matrix which consist of $user_i \times input_i$ as seen in figure 4. We will have a deeper look into the rating matrix in the chapter Methodology and Design.

	input1	input2	input3	input 4	input 5
user1	?	5	1	?	5
user2	2	4	?	2	?
user3	5	?	1	?	?
user4	1	?	?	5	2
user5	?	?	1	?	5

Figure 4: Recommended products with mixed reviews (one to five stars and unkown ratings), see appendix preRatingMatrixBasic.png on microSD

There are three standard PRE approaches which are collaborative filtering (CF), content-based filtering (CBF), and hybrid-based filtering (HBF).

2.2 Content-based filtering

Geetha mentions in [13] that CBF uses the historical purchase data of a customer, and it needs information about the features such as name, material, brand, production place, genre, etc. The system does not need to know the characteristics of a customer [14]. A consumer profile is created through the features of past purchases. The more the customer buys, the more detailed the profile will be, leading to higher classification accuracy. As shown in figure 5, the PRE classifies a second product similar to the purchased product. Therefore, the system recommends it to the client.

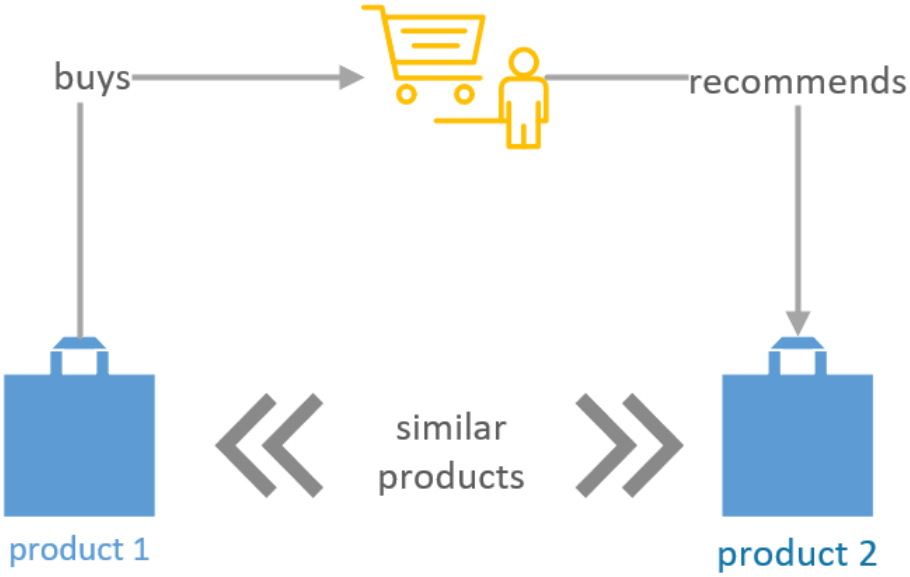


Figure 5: content-based filtering, in style of [13], see appendix preContentBased.png on microSD

In the following example, we have a content-based rating matrix in figure 6. The books ($books_i = \{book_1, book_2, ..., book_n\}$) have the attribute drama or thriller. The ratings of each are from one to five where one is terrible and five is very good. Some users ($user_i = \{user_1, user_2, ..., user_n\}$) don't have ratings for certain books yet.

	user1	user2	user3	user4	user5	Thriller	Drama
book1	?	5	1	?	5	yes	no
book2	5	4	?	2	?	yes	no
book3	2	2	1	?	?	no	yes
book4	4	?	?	5	2	yes	yes
book5	1	1	1	?	5	no	yes
book6	?	3	1	?	5	yes	no

Figure 6: Rating matrix for content-based filtering, see appendix preContentBasedTable1.png on microSD

In Figure 7, user2 has rated book1, book2, book3, and book4. At the same time, user1 has also rated book2 to book5. Based on the attribute thriller, the CBF will recommend user1 book1 because book1 is a book with the attribute 'thriller' and has the highest rating of all thriller books on the list.

	user1	user2	user3	user4	user5	Thriller	Drama
book1	?	5	1	?	5	yes	no
book2	5	4	?	2	?	yes	no
book3	2	2	1	?	?	no	yes
book4	4	?	?	5	2	yes	yes
book5	1	1	1	?	5	no	yes
book6	?	3	1	?	5	yes	no

Figure 7: Rating matrix for content-based filtering based on attribute thriller, see appendix preContentBasedTable2.png on microSD

2.3 Collaborative filtering

Collaborative Filtering is the most common technique for building intelligent recommender systems that can learn to give better recommendations as more information about users is collected [15]. Amazon and Spotify have been using the method for more than a decade. Unlike in CBF, CF needs to know the characteristics of a customer. The basic foundations of CF are to filter out which items a user will like based on the interactions of similar users.

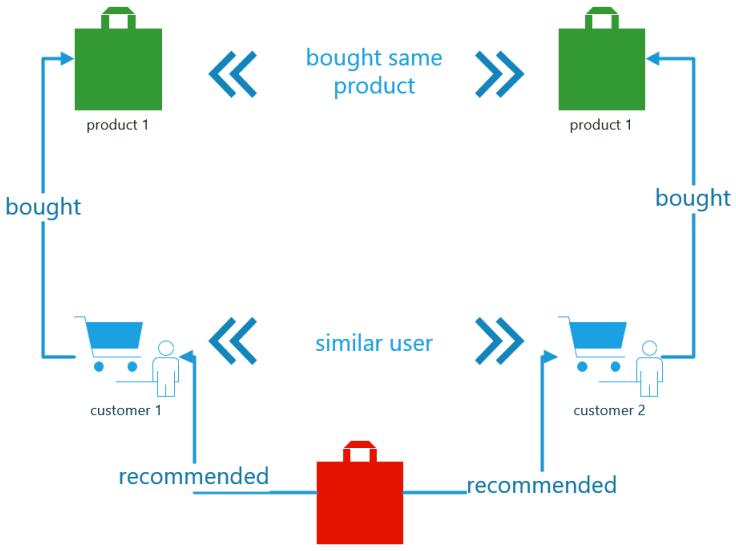


Figure 8: Model of product recommendation engine based on collaborative filtering, in style of [13], see appendix preCollaborative.png on microSD

In figure 8, customer1 and customer2 both bought the same product1. Due to the similarity of both users, the PRE recommends a second product to both customers. CF depends on the history of user-item interactions to provide recommendations as precisely as possible. These kind of interactions includes user-item, user-user or item-item correlations [13], [16]. This kind of filtering is known as memory-based filtering. We will look deeper into this topic in the chapters Methodology and Design. As shown in figure 9, CF uses two different techniques, model-based and memory-based. Model-based filtering commonly uses ML algorithms like clustering such as K-Nearest Neighbours (KNN) to find the k nearest neighbor to a user or item based on the similarity metrics used.

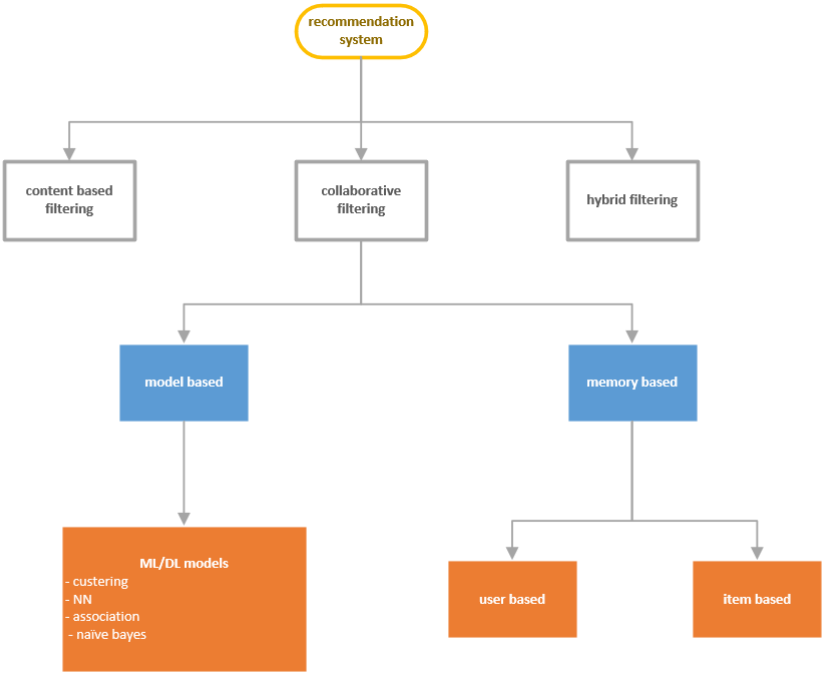


Figure 9: Model of common product recommendation engines, in style of [13], see appendix preModels.png on microSD

2.4 Hybrid-based filtering

Hybrid-based filtering is a technique that combines CBF and CF approaches. According to Geetha in [13], HBF has different approaches to utilizing both filtering systems:

- uniting the prediction results of both CBF and CF (figure 10)
- adding CBF to CF
- combine both approaches CBF and CF

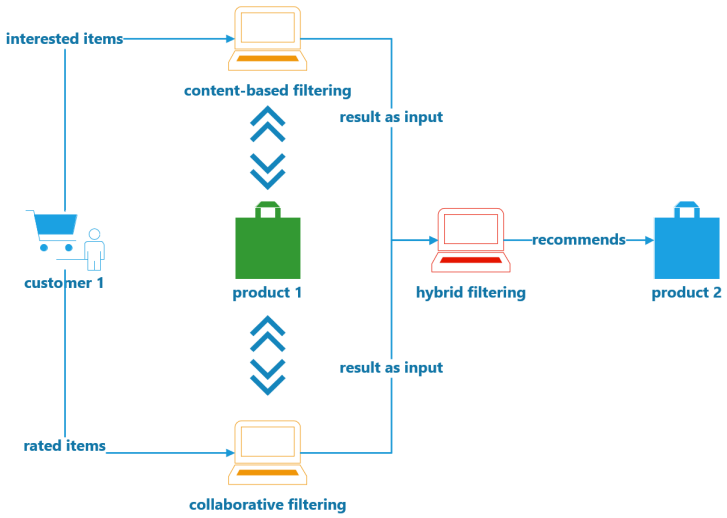


Figure 10: Model of common hybrid-based filtering system., see appendix preHybridBased.PNG on microSD

In various papers like [17], [18], researchers found that HBF can significantly increase prediction accuracy. It is mentioned that HBF can solve two problems, cold start, and sparsity, which are often occurring in PRE. When a new item or a new user is added to the system, and no user-item interaction has emerged yet, we face the problem of a cold start [19]. Cold start (figure 11) is a problem that CF faces since its recommendation depends on a certain degree of interactions between item-item, user-item, or user-user. In the case of sparsity, CF is facing "the difficulty in finding sufficient reliable similar users since in general the active users only rated a small portion of items" [20].

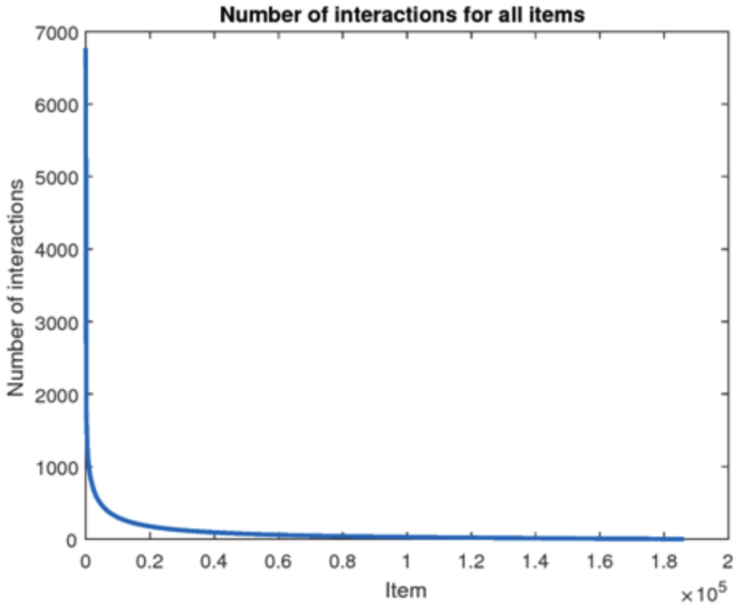


Figure 11: The graph shows that a small amount of items have a high interactions between users, while a great amount of items have either little or no interactions [21], see appendix preCollabColdStart.png on microSD

The streaming platform is an excellent example of using HBF effectively. Netflix compares the watching and search history and combines these data (CF) with movies or tv shows with similar features (CBF). In addition, data from user-item interactions (CF) such as movies and TV shows that other users most watch also play a role in the PRE.

There are several established HBF approaches [13] like:

- **Weighted prediction:** Certain recommendation elements in the predictions are weighted higher
- **Mixed predictions:** Combine predictions of various approaches to one recommendation.
- **Feature recommendation:** The output of a computed feature or a set of feature is used as input for further predictions.
- **Feature combination:** Different features from different knowledge sources are combined as input to one prediction.

3 Methodology and Design

To understand how to implement a PRE based on HBF, we will have a deeper look into how we can utilize matrix factorization on products with known or unknown features. We will also conduct a survey regarding user opinions about product recommendations and analyze the result in the chapter conclusion.

The following subchapters, 3.1.1 to 3.1.3, are based on the videos of Serano Academy [22], [23].

3.1 Concepts of collaborative filtering

In the previous chapter 2.3, we learned that CF recommendations are based on interactions and similarities between user-item, user-user, or item-item. In this subchapter, We will concentrate on the functionalities of user-item-based filtering by using single value decomposition (SVD) which is based on matrix factorization (MTF).

3.1.1 What is a user-item matrix?

Before we get into MTF, let's look at the basics of how user-item filtering works. We highlighted the same rating numbers with the same colors to make it easier to see the differences. In figure 12, we see a user-item matrix where all users have the same interests.

	I1	I2	I3	I4	I5
Alice	2	2	2	2	2
Bob	2	2	2	2	2
Eve	2	2	2	2	2
Dave	2	2	2	2	2

Figure 12: User-item matrix where all users have the same interests, in style of [22], see appendix preMaFactorDepSame.png on microSD

Every item with a correlation to a user is rated two. In figure 13, we have a user-item matrix where none of the users have the same interest, and none of the user-item correlations has the same rating.

	I1	I2	I3	I4	I5
Alice	1	2	3	4	5
Bob	2	1	5	3	4
Eve	4	5	1	2	3
Dave	5	3	4	1	2

Figure 13: User-item matrix where all users have different interests, in style of [22], see appendix preMaFactorDepDif.png on microSD

Now let's look at our third matrix example in figure 14. Compared to both matrices in figure 12 and figure 13, the matrix in figure 14 is a more realistic approach, and it includes similarities between users and items.

	I1	I2	I3	I4	I5
Alice	3	1	1	3	1
Bob	1	2	4	1	3
Eve	3	1	1	3	1
Dave	4	3	5	4	4

Figure 14: User-item matrix where some users have similar interests, in style of [22], see appendix preMaFactorDepRealistic.png on microSD

In figure 15, we have Alice and Eve, who rated the items $I_i = I_1, ..., I_5$ with the same ratings. In correlation with the user-item, we can interpret it as Alice and Eve having similar interests in correlation to the user-item in correlation with the user-item.

	I1	I2	I3	I4	I5
Alice	3	1	1	3	1
Bob					
Eve	3	1	1	3	1
Dave					

Figure 15: User-item matrix where Alice and Eve have similar interests, in style of [22], see appendix preMaFactorDepRealisticSim1.png on microSD

The table in figure 16 appears not to include similarities between users or items. But when we look closely at the ratings of Bob, Eve, and Dave, we observe that the ratings of $Dave = Bob + Eve$. For example, the sum of ratings made by Eve and Bob for Item I_1 is four, and that means Dave's interests are a combination of Bob and Eve.

	I1	I2	I3	I4	I5
Alice					
Bob	1	2	4	1	3
Eve	3	1	1	3	1
Dave	4	3	5	4	4

Figure 16: User-item matrix where Bob and Eves ratings combined together results in the ratings of Dave, in style of [22], see appendix preMaFactorDepRealisticSim2.png on microSD

A more practical example would be the statements: "Bob loves PC with face unlock", "Eve loves PC with biometric unlock", "Dave loves PC with biometric unlock", and "Dave loves PC with face unlock and biometric unlock". In figure 17, we have the items I_1 and I_2, which the users equally rate. It does not mean, that both items are exactly the same. It means, both items have similar features. Features such as shown in figure 18.

	I1	I2	I3	I4	I5
Alice	3			3	
Bob	1			1	
Eve	3			3	
Dave	4			4	

Figure 17: User-item matrix where I_1 and I_2 have equal rating through out each user, in style of [22], see appendix preMaFactorDepRealisticSim3.png on microSD

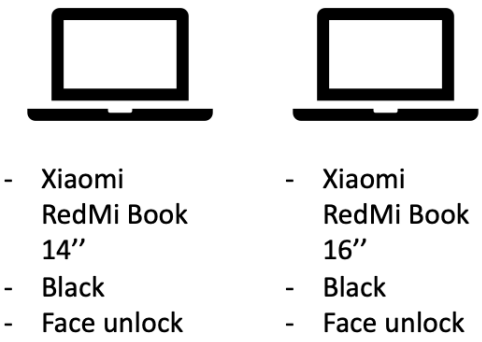


Figure 18: I_1 and I_4 have similar features, in style of [22], see appendix preMaFactorRealScenario.png on microSD

In figure 16, we notice that the ratings of $I_5 = avg(I2, I3)$, meaning I_5 has features of both I_2 and I_3 (figure 2). In figure 21, not all users rate their bought items in the real world; therefore, a rating of such items is missing. These correlations between items and users are essential for the PRE. Once established, products could be recommended to users who do not own these items.

	I1	I2	I3	I4	I5
Alice		1	1		1
Bob		2	4		3
Eve		1	1		1
Dave		3	5		4

Figure 19: User-item matrix where the ratings of I_5 is the average of I_2 and I_3 ratings, in style of [22], see appendix preMaFactorDepRealisticSim4.png on microSD

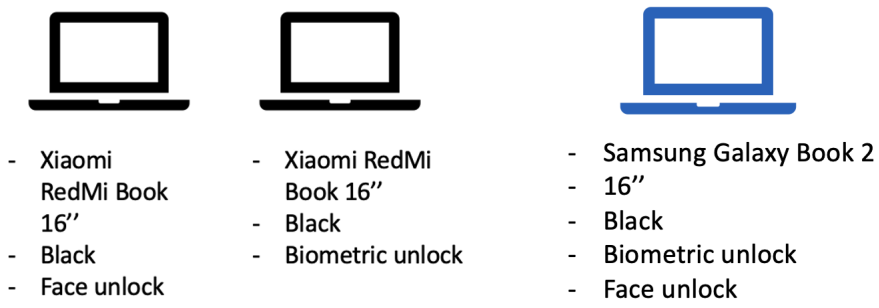


Figure 20: I_5 (here Samsung Galaxy Book) combines both features of I_2 and I_3, in style of [22], see appendix preMaFactorRealScenario2.png on microSD

	I1	I2	I3	I4	I5
Alice	3	?	1	3	1
Bob	1	2	4	?	3
Eve	?	1	1	?	?
Dave	4	3	5	4	4

Figure 21: I_5 (here Samsung Galaxy Book) combines both features of I_2 and I_3, in style of [22], see appendix preMaFactorRealisticPartial.png on microSD

3.1.2 What is matrix factorization?

When it comes to factorization, we usually think of an equation like $8 * 4 = 32$. In MTF, we break down a given matrix into two feature matrices. In figure 22, we have an item feature matrix (a) and an user feature matrix (b). The item feature matrix includes two features, biometric unlock and face unlock. We assign rating numbers to each feature for each item. For example I_1 has the feature biometric unlock, but not face unlock. We know from figure 21, that Alice and Dave love I_1, so we calculate the average of their ratings for I_1. The result would be $biometricUnlock = avg(I_1(Alice), I_1(Dave)) \approx 3$. For face unlock, we have the ratings by Bob and Eve. Eve has not rated I_1 yet. Therefore we set $Eve = 0$ and calculate $faceUnlock = avg(I_1(Bob), I_1(Eve)) \approx 1$.

	biometric unlock	face unloc
I1	3	1
I2	1	2
I3	1	4
I4	3	1
I5	1	3

(a) item feature matrix

	biometric unlock	face unlock
Alice	1	0
Bob	0	1
Eve	1	0
Dave	1	1

(b) user feature matrix

Figure 22: feature matrices, in style of [22], see appendix preFeatMaItem.png & preFeatMaUser.png on microSD

The user feature matrix includes the features biometric unlock and face unlock. Our numbers here stand for TRUE or FALSE. For example Alice likes biometric unlock, but she does not like face unlock. We can perform the dot product $\sum_{i=1}^{n} a_i b_i = I_i(user_i) = itemRating$ with these two feature matrices. The calculated rating allows the PRE to know which item a user would most likely buy and therefore should recommend to the user. Figure 23 shows an example of MTF where the user-item matrix is missing values (a) and the user-item matrix after the dot product was performed (b). Eves rating for I_4 is $I_4(Eve) = 3 * 1 + 1 * 0 = 3$ and for I_5 is $I_5(Eve) = 1 * 1 + 3 * 0 = 1$. Knowing the rating values, a PRE would recommend I_4 to Eve.

Matrix factorization

	I1	I2	I3	I4	I5
biometric	3	1	1	3	1
face	1	2	4	1	3

	biometric	face
Alice	1	0
Bob	0	1
Eve	1	0
Dave	1	1

	I1	I2	I3	I4	I5
Alice	3	?	1	3	1
Bob	1	2	4	?	3
Eve	?	1	1	?	?
Dave	4	3	5	4	4

(a) matrix factorization with missing rating values;

Matrix factorization

	I1	I2	I3	I4	I5
biometric	3	1	1	3	1
face	1	2	4	1	3

	biometric	face
Alice	1	0
Bob	0	1
Eve	1	0
Dave	1	1

	I1	I2	I3	I4	I5
Alice	3	1	1	3	1
Bob	1	2	4	1	3
Eve	3	1	1	3	1
Dave	4	3	5	4	4

(b) calculated user feature matrix after using MTF

Figure 23: MTF results [23],see appendix preFeatMaDotProduct1.png & preFeatMaDotProduct2.png on microSD

One important thing to know is, throughout this subsection, we always know which features an item have. In the real world, a lot of retailers do not register which features their products have on e-commerce platforms like Amazon. The same applies to the user-item rating matrix. In our examples most items were rated by the users. It does not include the cold start problem as well as most users do not give ratings to items they bought. We will look into how to perform MTF on products with missing features and ratings in the later chapters.

3.1.3 Concepts of single value decomposition

Our data set (more in later chapters) is a high-dimensional matrix and it has ten thousands of input variables. This high dimensionality comes with a problem. The authors in [24] mentions that high-dimensional functions can be more complicated than low-dimensional ones. A good solution is to use single value decomposition which is a method of MTF using the technique dimensional reduction $Loss = \Sigma_i \Sigma_j (A_{ij} - u_i {u_j}^T)^2$. A_{ij} are the original values of the matrix and $u_i u_j$ are the values generated by the algorithm. T stands for transpose (dimension). The calculated values can also be negative, for this reason, we need to square our result. Dimensional reduction reduces the number of inputs in a given training data. In figure 24, we have a 4x4 matrix that is reduced to a 4x1 matrix (u_i) and a 1x4 matrix (u_j).

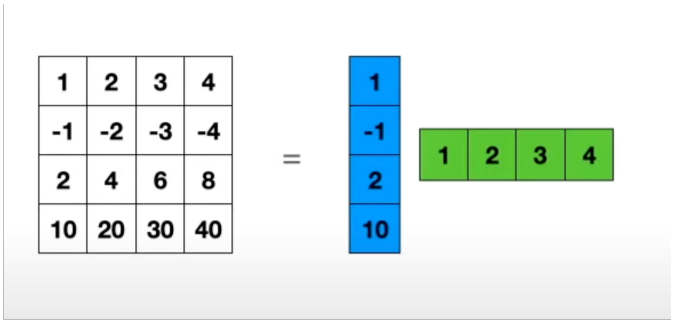
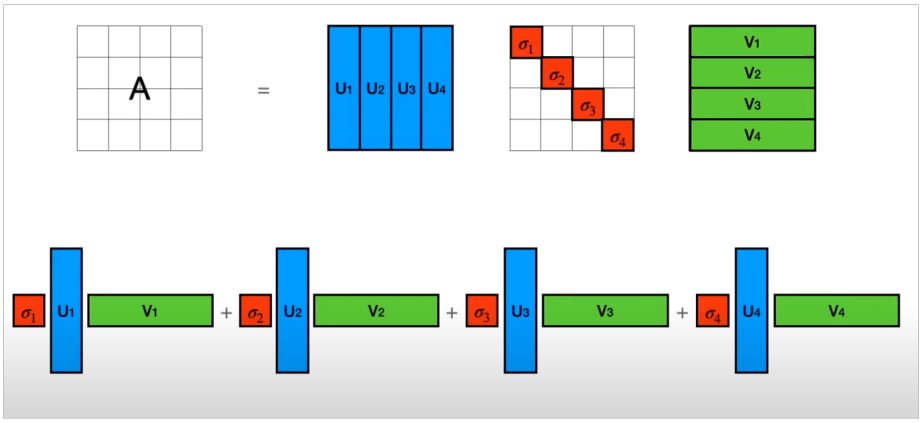


Figure 24: reduced 4x4 matrix of [23], see appendix svdDimReducSimple.png on microSD

Dimensional reduction in SVD is slightly different. We have a $R = U\Sigma V$ where R is the calculated rating, U are the values of each column, Σ stands for dimension as well as weight, and V are the values of each rows [14]. The 4x4 matrix in figure 25 (a) is broken down into its columns $\{u_1, u_2, u_3, u_4\}$, rows $\{v_1, v_2, v_3, v_4\}$, dimension and weight $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. In cases where weight is not needed, we can set $\sigma_1 ... \sigma_4 = 1$. Same as our previous examples in MTF, we also perform dot product here. The results are summed up after each iteration.



(a) SVD dimensional reduction for 4x4 matrix

(b) example for SVD dimensional reduction for 4x4 matrix

Figure 25: SVD dimensional reduction [23], see appendix svdDimReduc.png & svdDimReduc2.png on microSD

In both examples (figure 24 and figure 25), we have a SVD with rank 1. For these two, we can also have rank 1 to rank 4. Generally speaking the number of ranks depend on the number of dimensions (figure 26). A 5×5 matrix would have 5 ranks and a $n \times n$ matrix would have n ranks. But these rules only apply to squared matrices. SVD handles dimensional reduction differently when we have non-squared matrices which is most likely in a real world scenario. There will be always more products than users.

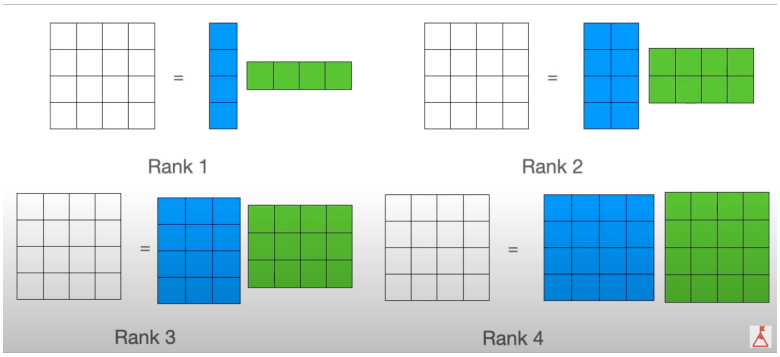


Figure 26: ranks in dimensional reduction, in style of [23], see appendix svdDimReducRank.png on microSD

We have a 4×6 matrix in figure 27. Like previous examples, we have 4 columns $U = \{u_1, ...u_4\}$. Similar to figure 25, we use the first four column as dimension and weight $\Sigma = \{\sigma_1, ..., \sigma_4\}$, the last two columns will be padded with zeros and converted into rows. That's the reason why we have six rows $V = \{v_1, v_2, ..., v_6\}$ instead of four.

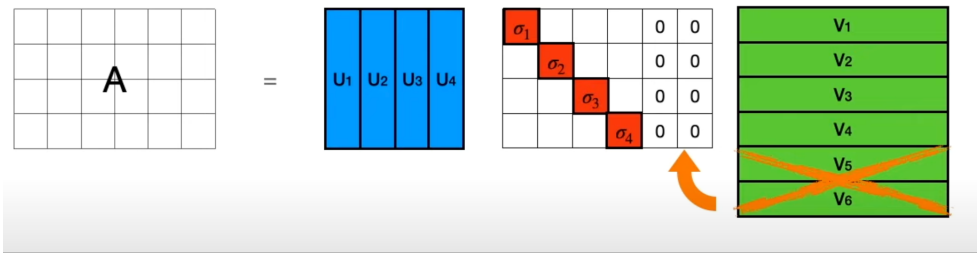


Figure 27: SVD for non-square matrices, in style of [23], see appendix svdDimReducNoSquare.png on microSD

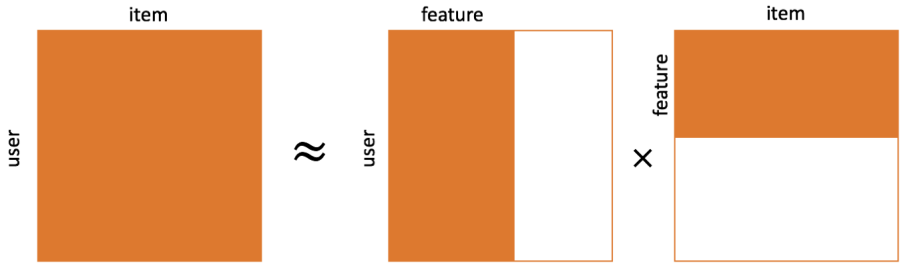
3.2 Single value decomposition on products with unknown features or missing features and ratings

This chapter is based on [25].
Our data set will have a mix of products with clear features and products with missing or unknown number of features and ratings. In this case, SVD works slightly different. In figure 28 a, we have a user-item matrix where users (M) and items (N) are assigned an unique id. The value 0 stands for unknown rating and 1 is the lowest rating an user can give an item. With MTF, we can find our k latent features. As in chapter 3.1.3, we will decompose the matrix in two matrices, user-feature $(M \times K)$ and item-feature $(N \times K)$. Therefore in figure 28 b, we have $R = W\Sigma Z^T \Leftrightarrow$

$R(M \times N) = W(M \times K)\Sigma(K \times K)Z(N \times K)^T$ which is similar to the equation $R = U\Sigma V$. We use MTF to remove features which are non-relevant to user ratings. This way, we can find our hidden features and reduce unnecessary noise. We can calculate the rating (r_{ui}) by an user (w_{uk}) for an item (z_{ki}) across all unknown features k by using dot product $r_{ui} = \sum_{k=1}^{K} w_{uk}\sigma_k z_{ki}$.

	I1	I2	I3	I4	I5
Alice	3	5	4	4	2
Bob	0	0	0	0	0
Eve	0	0	0	0	0
Dave	4	3	0	0	0

(a) User-item table with unknown features and ratings



(b) Dimensional reduction for matrices with unknown features

Figure 28: MTF in SVD with unkown features and ratings [25], see appendix svdUnkownFeatExampleTable.png & svdUnkownFeat.png on microSD

This is how we can receive ratings from known items and users. The question is now what happens if a new user is added to the system? The structure of feature weight matrix Σ and the item-feature matrix Z won't change, but the user-feature matrix W will. We have $R = W\Sigma Z^T$. If we multiply Z on each side, we have $RZ = W\Sigma Z^T Z$ and $Z^T Z = 1$. Therefore we can simplify the equation to $RZ = W\Sigma$. The new user-feature matrix will be calculated by using $RV\frac{1}{\Sigma} = W$. In case of a new item is added to the system, we can use the same methods for the Z matrix.

Now we know how to obtain both matrices, user-feature and item-feature, we can use gradient descent to minimize the squared error (e_{ui}) between the actual ratings (r_{ui}) and the predicted ratings $(\widetilde{r_{ui}})$. We can calculate squared error by using $e_{ui}^2 = (r_{ui} - \widetilde{r_{ui}})^2 = (r_{ui} - \sum_{k=1}^{K} w_{uk}\sigma_k z_{ki})$. After each iteration, we need to update the rating matrix. First we need to use gradient descent on both matrices, $\frac{\partial}{\partial w_{uk}} e_{ui}^2 = -2(r_{ui} - \widetilde{r_{ui}})z_{ki} = -2e_{ui}z_{ki}$ for user-feature and $\frac{\partial}{\partial z_{ki}} e_{ui}^2 = -2(r_{ui} - \widetilde{r_{ui}})w_{uk} = -2e_{ui}w_{uk}$ for item-feature.

Knowing the gradients allows us to update our ratings after each iteration, minimizing our error margin. We also add a learning rate α which decides the size of our updates. The update rules for user-feature matrix is $dw_{uk} = w_{uk} - \alpha^* \frac{\partial}{\partial w_{uk}}(e_{ui}^2) = w_{uk} + 2e_{ui}z_{ki}$ and for item-feature matrix is $dz_{ki} = z_{ki} - \alpha^* \frac{\partial}{\partial z_{ki}}(e_{ui}^2) = z_{ki} + 2e_{ui}w_{uk}$. Once we get the optimal user-feature and item-feature matrices, we can start predict the ratings in our PRE.

3.3 Designing a recommendation system using SVD

Before we design our PRE, we will have a first look at our data set. Our data set has 4.2 million unique users and 476001 unique items (figure 29). It is a data set from Kaggle [26] which is a parsed data set by the authors of [27], [28]. All items have an average rating, but not all users have rated all items. The minimum rating a user can give to an item is 1.0 (one star), and the highest is 5.0 (five stars). The data set is unbalanced. We have above-average ratings of five stars compared to one to four stars.

Detail Compact Column 4

AKM1MP6P0OYPR	0132793040	# 5.0	# 1365811200
4201696 unique values	**476001** unique values	1 5	913m 1.41b
A2CX7LUOHB2NDG	0321732944	5.0	1341100800
A2NWSAGRHCP8N5	0439886341	1.0	1367193600
A2WNBOD3WNDNKT	0439886341	3.0	1374451200
A1GI0U4ZRJA8WN	0439886341	1.0	1334707200

Figure 29: Data set for implemenation [26], see appendix designDataSet.png on microSD

For the implementation (figure 30), we will first read our data set and prepare it for further processing. Then we will create a data frame by filtering our item ratings with four to five stars, leading us to our rating matrix, which we will be using for MTF. We will fill unknown user-item ratings with 0.0 to prevent data type inconsistency. The next step is to use dimensional reduction with SVD and gradient descent to get the correlation matrix. For the last step of the prediction, we will choose a randomized user and recommend the top ten products with four to five stars.

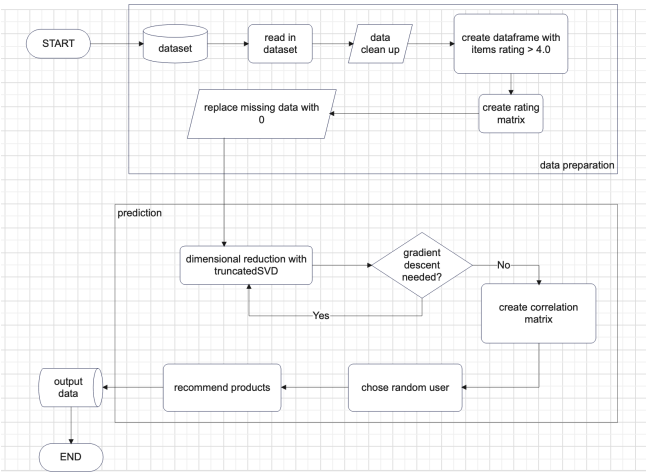


Figure 30: Implementation flow chart, see appendix designProcessFlowchart.png on microSD

4 Implementation

In this chapter, we have a deeper look in how to implement a PRE which only recommends products with four to five stars. Each stage has a different functionality, but it is build on top of each other. All implementation showed in this section are just a fraction of what we actually implemented into the PRE.

4.1 Stage 1: Choosing the right tools

Choosing the right tools is essential for the implementation. Due to the large data set a high memory usage has to be considered. We looked into three standard tools used for ML implementations.

- **Microsoft Visual Studio Code:** Visual Studio Code is Microsofts own multipurpose IDE, which is very versatile for many languages. However, it has some flaws, especially in displaying tables or charts used in standard ML libraries like matplotlib.
- **Jupyter Notebook:** Jupyter Notebook is the most used open-source scripting tool in ML. Compared to Microsoft Visual Studio Code, it displays plots and has a clear structure for scripting. It uses about 900 MB of RAM on MacOS. If we add the base MacOs system usage, we would have 3.7 GB / 16 GB of RAM that we can not use for our calculation.
- **Google Colab Notebook:** Google Colab Notebook is Google's version of the Jupyter Notebook. It has all the advantages of the Jupyter Notebook, and Additionally, it has two runtime environments. We can either use our computer or choose Google's runtime environment for Google Colab Notebook, which is managed server-side.

We decided to use Google Colab Notebook. With the possibility to use Googles own runtime environment, we can fully utilise its 16GB RAM without sacrificing memory for our systems own environment usage or other applications running in the background.

4.2 Stage 2: Prepare data set

As mentioned in chapter 3.3, our data set has 4.2 million unique users and 476001 unique items. Due to hardware limitations, we will not be able to use all of the data. We first looked into our data set (figure 31 a), retrieved the data (figure 31 b), and calculated the average, minimum, and maximum item ratings (figure 31 c). We filtered our data by items with ratings above 4.0.

```
df = pd.DataFrame(data)
df = df.loc[df['Rating'].isin([4, 5])]
```

Due to our hardware limitations, we must create a utility matrix by choosing a fraction of our data (figure 32). This matrix will be used for dimensional reduction with SVD.

```
data1 = df.head(100000)
ratings_utility_matrix = data1.pivot_table(values='Rating',
index='UserId', columns='ItemId')
```

All missing values will be filled with 0.0 to keep the data consistent.

	UserId	ItemId	Rating	Timestemp
0	A2CX7LUOHB2NDG	0321732944	5.0	1341100800
1	A2NWSAGRHCP8N5	0439886341	1.0	1367193600
2	A2WNBOD3WNDNKT	0439886341	3.0	1374451200
3	A1GI0U4ZRJA8WN	0439886341	1.0	1334707200
4	A1QGNMC6O1VW39	0511189877	5.0	1397433600

(a) First five rows of raw data set

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 4 columns):
 #   Column     Non-Null Count    Dtype
---  ------     --------------    -----
 0   UserId     1048575 non-null  object
 1   ItemId     1048575 non-null  object
 2   Rating     1048575 non-null  float64
 3   Timestemp  1048575 non-null  int64
dtypes: float64(1), int64(1), object(2)
memory usage: 32.0+ MB
```

(b) Raw data set information

	Rating	Timestemp
count	1.048575e+06	1.048575e+06
mean	3.973379e+00	1.248822e+09
std	1.399329e+00	1.091615e+08
min	1.000000e+00	9.127296e+08
25%	3.000000e+00	1.169078e+09
50%	5.000000e+00	1.250035e+09
75%	5.000000e+00	1.355789e+09
max	5.000000e+00	1.406074e+09

(c) Average, minimum and maximus of given ratings

Figure 31: Prepare raw data set for processing, see appendix implRawData.png, implRawDataInfo.png, and implRawDataAvg.png, on microSD

UserId	A00766851QZZUBOVF4JFT	A00995931BE16NG4F52QC	A01255851ZO1U93P8RKGE	A014623426J5CM7M12MBW	A01852072Z7B68UHLI5UG	A0266076X6KPZ6CCHGVS	A0293130VT
ItemId							
0321732944	0.0	0.0	0.0	0.0	0.0	0.0	
0511189877	0.0	0.0	0.0	0.0	0.0	0.0	
0528881469	0.0	0.0	0.0	0.0	0.0	0.0	
059400232X	0.0	0.0	0.0	0.0	0.0	0.0	
0594012015	0.0	0.0	0.0	0.0	0.0	0.0	

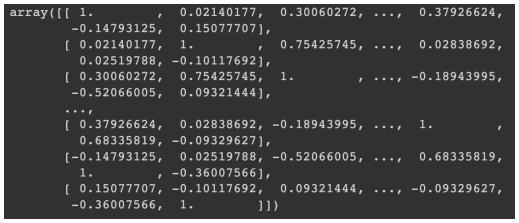
Figure 32: Prepare utility matrix for user-item ratings which will be used for dimensional reduction, see appendix implUtilityMaRating.png on microSD

4.3 Stage 3: Prediction phase

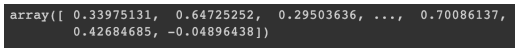
In this phase, we will start our prediction of whether an item can be recommended to a user or not. We decomposed our previously created utility matrix using truncatedSVD and created a correlation matrix which gave us information about the similarities between users and items (figure 33 a). Negative numbers mean no similarities were found. Furthermore, positive numbers mean that some similarities were found. The closer a value is to 1.0, the greater the similarity. In addition, we chose a random user and found items that similar users have rated (figure 33 b) and recommended items with a similarity above 0.90 (figure 33 c).

```
correlation_ItemId = correlation_matrix[i]
```

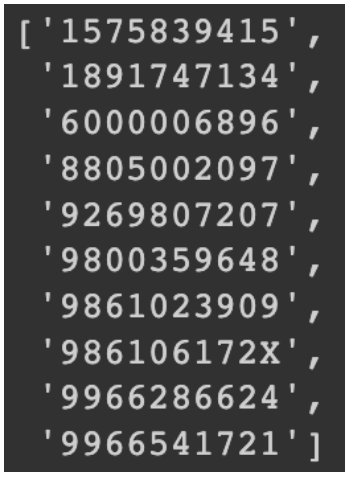
```
correlation_ItemId
recommended_items = X.index[correlation_ItemId > 0.90]
recommended_items = list(recommended_items)
recommended_items[0:10]
```



(a) Decomposed matrix for similarities



(b) Similar items for a specific user

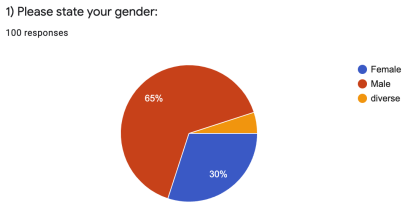


(c) Prediction result

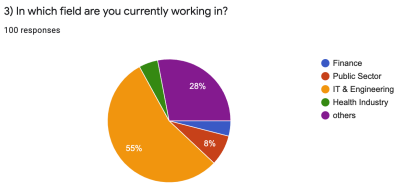
Figure 33: Similarity matrix results, see appendix implUserItemSim.png, implRandUserItemSim.png, and implPredResult.png, on microSD

5 Survey results

As mentioned in the chapter "Method and Design," we conducted a survey to determine how consumers think about product recommendations. Our target group is everyone above 18 years old; therefore, everyone is legally allowed to fulfill a purchase contract without a legal representative (§106 BGB).



(a) Users gender



(b) Users career

(c) Users visit online shops

(d) Users buy product

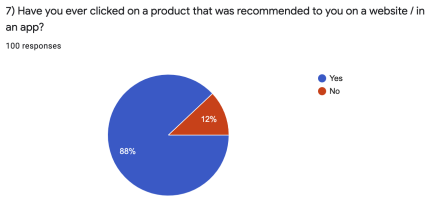
Figure 34: Overview of survey participants and users shopping behavior, see appendix surveyGender.png, surveyCareer.png, surveyShopBehav2.png, and surveyShopBehav3.png on microSD

One hundred users participated in the survey. 65% were male, 30% were female, and 5% were diverse (figure 34 a). The high male participation mirrors the career path "IT and Engineering," which is 55% and the primary career choice of our participants (figure 34 b). We learned in the question section " user shopping behavior" that most users, 51%, visit an online shop every week (figure 34 c). However, only 21% buy a product every week, and the majority of the users, 53%, buy an item at least once a month (figure 34 d).

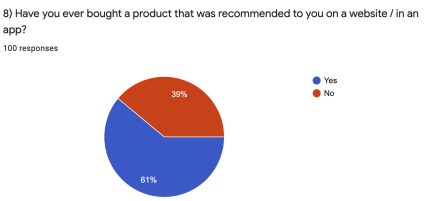
Regarding whether a customer has ever clicked on a recommended product on a website or in an app, 88% answered with yes (figure 35 a). However, 61% bought a recommended product (figure 35 b). In figure 35 c, most participants (40%) are neutral towards product recommendations. 24% found it not helpful and 4% absolutely not helpful. Compared to these, 32% were more appealed to product recommendations, where 27% thought it is helpful and 5% very helpful.

The participants were instructed to imagine buying new in-ear headphones. They should choose between headphones with consistent four to five stars ratings and headphones with diverse ratings like zero to five stars. 69% chose four to five stars (figure 35 d). The people who chose diverse ratings stated the following answers (selected few. For all statements, see appendix: column number 10.2) "Please give a short answer why you would..." in preSurveyResponses.xlsx):

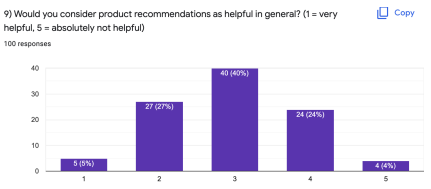
- I trust in review by real users.
- It is an opportunity to investigate further by myself.
- It's unrealistic no one has had issues. I like reading only the bad comments about a product.
- Diverse ratings, diverse experiences = find you own
- Find it hard to believe that a product can be that good so the range from 0-5 feels much more realistic.



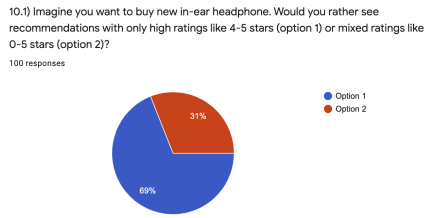
(a) Users clicked on recommendations



(b) Users bought recommendations



(c) Helpfulness of recommendations



(d) Best or diverse product ratings

Figure 35: Results regarding consumer interactions with product recommendations and what they think about the usefulness of recommendations, see appendix surveyProdRecClick.png, surveyProdRecBuy.png, surveyProdRecHelpful.png, and surveyProdRecBestDiverse.png, on microSD

The consumers were given a picture (figure 36 a) which shows five in-ear headphones with ratings from one to five stars. The scenario is that the online shop has upgraded its PRE. Consumers should now consider whether product recommendations with four to five stars ratings are better than product recommendations with three to five stars ratings. 58% considered product recommendations with four to five stars ratings as an improvement compared to one to five stars ratings (figure 36 b).

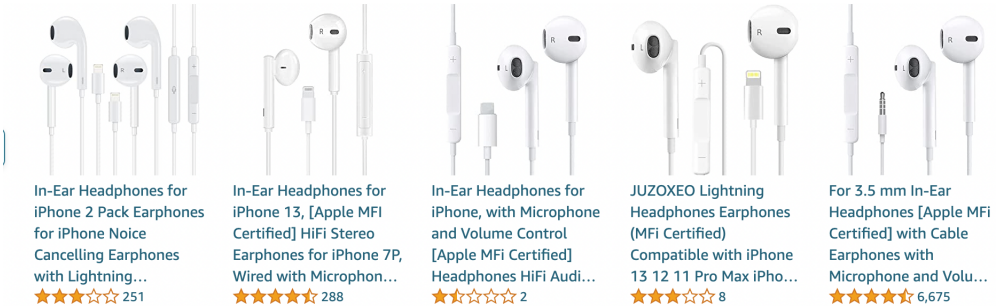
The participants were asked to give a short answer regarding why they chose four to five stars ratings or three to five stars ratings (selected few. For all statements, see appendix: column numbers 11.2) [four to five stars] and 11.3) [three to five stars] in preSurveyResponses.xlsx).

Four to five stars ratings:

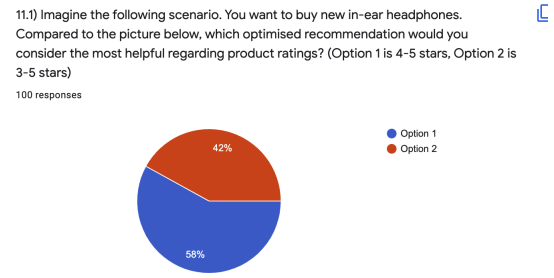
- Higher ratings mostly leads me to the conclusion that the seller is reliable and I won't face problems while buying it.
- Because they're probably better anyway and I dont habe to look through bad headphones.
- Products that have legitimate higher ratings help avoid lesser quality products.
- I want a product which performs above average.
- I'm not interested in low rated products as it normally represents poor user experience.

Three to five stars ratings:

- More authentic.
- I would like to know why people dislike it an see if that a big deal for me or not.
- Some 4 to 5 stars can be fake. I would rather see what others have to offer.
- Higher range of opinions are always better.
- Products are different: some people like them, some people don't. I rather decide on good and bad reviews and my own experience than just on the number of stars.



(a) Starting scenario for PRE optimisation



(b) Survey results whether four to five stars ratings or three to five stars ratings are an improvement to (a)

Figure 36: PRE improvements, see appendix surveyCompare1.png & surveyPREImprove.png on microSD

6 Conclusions

In this paper, we proposed recommending high-quality products rated four to five stars by users using an optimized PRE. We looked into different standard RE and their methods. Following our research, we decided to use SVD, a method of MTF, and combined it with gradient descent to reduce the error margin. We were able to recommend ten products to a randomly chosen user. These products are four to five stars and were determined through a similarity matrix.

Our survey results with 100 consumers showed us that 40% of the participants are neutral towards PRE, and 32% considered it helpful. Furthermore, 58% of the participants would consider product recommendations with four to five stars helpful, and they will help them through the purchase process. These customers consider higher product ratings to indicate excellent product quality and user-friendly experiences. Some customers even stated that high product ratings could be a time-saver while trying to find a suitable product for their usage.

However, it should be considered that a survey with 100 participants is not representative and does not mirror the overall society.

7 Lessons learned

During our research phase, we encountered a few difficulties. It was not easy to find unbiased data about e-commerce market development for the introduction. At the beginning of our research, many statistics came from e-commerce compa-

nies or e-commerce unions. These are often derived from the statistics from NGOs (non-government organizations) or organizations specializing in market development analysis. It was a challenge to distinguish biased data from unbiased data.

Writing the state-of-the-art part was always a fight between whether or not we should dive much deeper into the topic to show all available PRE concepts instead of just presenting to most common ones. Due to our limited time, we displayed the most used PRE filtering techniques.

Knowing that our time and resources are limited, it was not trouble-free to decide which method we would use to process our data. After a week, it was clear that using SVD and deep learning combined won't help us solve the cold start problem. Our data set for the implementation is not suited for unsupervised learning. When we decided to use SVD for our PRE, another problem came up. We realized understanding the concepts of a method does not automatically lead to having the skill to explain a complex, mathematical problem the easy way.

The struggle to find an excellent way to explain a complex concept was as much as the late realization that our data set had a hidden problem we had not considered. The results of our first implementation did not meet our expectations. It was way off-chart than it should have been. About 89% of our rating matrix was full of negative numbers. Negative numbers meant that no users or items had similarities. The data set had more than four million entries which are unbelievable that most users or items had no similarities to others. After further investigations, we found that about 90% of the listed items had no features or were missing features. The data set also had users who did not rate any products yet.

It was naive to think that the first 300 rows of a data set with more than four million entries are a good representation of the whole data set. Not checking whether there are items with no features or missing features was a mistake that cost us valuable time, which we could have used on other parts of the paper.

This paper taught us that having a good general knowledge of AI and its disciplines can be an advantage and a disadvantage. Good pre-knowledge helps with complex theories and choosing a suitable data set. However, despite the excellent knowledge, we are less open-minded. We believe an open mind is an essential virtue working on a scientific paper.

8 Future Work

E-commerce changed our way of consumption, and its revenue will continue to grow. Global sales are expected to reach 7.3 trillion US dollars by 2025 [29]. Most PREs today are based on user browser history, purchase history, and other customer product interactions. However, many brands like Addidas and Samsung use influencers to promote their products on social media like Instagram, TikTok, and Youtube. The trend goes to in-app purchases. Last year Instagram started to offer live shopping services [30]. This gives users a real-time shopping experience any time and anywhere.

To keep up with live shopping and continuous customer satisfaction, we must find new ways to recommend products. One possible way is to use deep learning to mimic the neural network of a brain. Features should not only be limited to product features like colors, size, or price. We should also consider features like referrer (how a consumer learned about the product), time spent on a product or product type, in which order a user searched the products, or the live location of a person. In the case of location, it is not meant that the person is currently in New York and the PRE recommends typical New York souvenirs. The idea is to recommend a product on the fly while the person is walking down the street. For example, the person is interested in doing a bike tour and walks on a street where an e-bike shop is located. Due to this knowledge, the PRE recommends this person e-bikes from the shop.

We believe live shopping is the next big step in e-commerce. A good PRE will be needed to keep customers satisfied.

List of Figures

References

[1] A. C. Rivera, M. Tapia-Leon, and S. Lujan-Mora, "Recommendation systems in education: A systematic mapping study," in *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)* (Á. Rocha and T. Guarda, eds.), pp. 937–947, Springer International Publishing, 2018.

[2] M. E. Jennex and L. Olfman, *Development Recommendations for Knowledge Management/ Organizational Memory Systems*, pp. 209–222. Springer US, 2001.

[3] B. Xiao and I. Benbasat, "E-commerce product recommendation agents: Use, characteristics, and impact," *MIS Quarterly*, vol. 31, pp. 137–209, 03 2007.

[4] Salesforce, "What is a retail product recommendation engine – and why do you need one?." Available at `https://www.salesforce.com/resources/articles/what-is-product-recommendation-engine/` (2022/04/29).

[5] BigCommerce, "Ecomerce - frequently asked questions." Available at `https://www.bigcommerce.com/articles/ecommerce/#ecommerce-frequently-asked-questions` (2022/04/29).

[6] E. Eroglu, "The changing shopping culture: Internet consumer behavior," *Review of Business Information Systems June 2014*, vol. 18, no. 1, pp. 35–40, 2014.

[7] M. A. Rahman, M. A. Islam, B. H. Esha, N. Sultana, and S. Chakravorty, "Consumer buying behavior towards online shopping: An empirical study on dhaka city, bangladesh," *Cogent Business & Management*, vol. 5, no. 1, p. 1514940, 2018.

[8] UNCTAD, "Covid-19 has changed online shopping forever, survey shows." Available at `https://unctad.org/news/covid-19-has-changed-online-shopping-forever-survey-shows` (2022/04/30).

[9] Statista, "E-comerce worldwide." Available at `https://www.statista.com/topics/871/online-shopping/#editorialPicks` (2022/04/30).

[10] N. Suisse and UNCTAD, "Covid-19 and e-commerce: Findings from a survey of online consumers in 9 countrieds." Available at `https://unctad.org/system/files/official-document/dtlstictinf2020d1_en.pdf` (2022/04/30).

[11] Tageschau, "Im virtuellen supermarkt." Available at `https://www.tagesschau.de/wirtschaft/lieferdienste-onlinehandel-corona-boom-101.html` (2022/04/30).

[12] E. Torres-Moraga, A. Vásquez-Parraga, and J. Zamora, "Customer satisfaction and loyalty: Start with the product, culminate with the brand," *Journal of Consumer Marketing*, vol. 25, pp. 302–313, 08 2008.

[13] M. P. Geetha, "Research on recommendation systems using deep learning models," 01 2022.

[14] A. Roy, "Introduction to recommender systems- 1: Content-based filtering and collaborative filtering." Available at https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421 (2022/04/30).

[15] A. Ajitsaria, "Build a recommendation engine with collaborative filtering." Available at https://realpython.com/build-recommendation-engine-collaborative-filtering/ (2022/04/30).

[16] A. Biswas, K. S. Vineeth, A. Jain, and Mohana, "Development of product recommendation engine by collaborative filtering and association rule mining using machine learning algorithms," in *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*, pp. 272–277, 2020.

[17] G. Lekakos and G. Giaglis, "A hybrid approach for improving predictive accuracy of collaborative filtering algorithms," *User Model. User-Adapt. Interact.*, vol. 17, pp. 5–40, 02 2007.

[18] L. de Campos, J. Fernández-Luna, J. Huete, and M. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks," *Int. J. Approx. Reasoning*, vol. 51, pp. 785–799, 09 2010.

[19] X. Zhao, *Cold-Start Collaborative Filtering*. PhD thesis, London's Global University, 2016.

[20] G. Guo, "Improving the performance of recommender systems by alleviating the data sparsity and cold start problems," pp. 3217–3218, 08 2013.

[21] MaurizioFD, "Cold start (recommender systems)." Available at https://en.wikipedia.org/wiki/Cold_start_(recommender_systems) (2022/05/23).

[22] Serano.Academy, "How does netflix recommend movies? matrix factorization." Availble at https://www.youtube.com/watch?v=ZspR5PZemcs (2022/05/13).

[23] S. Academy, "Singular value decomposition (svd) and image compression." Availble at https://youtu.be/DG7YTlGnCEo (2022/06/13).

[24] R. O. Duda and P. E. Hart, "Pattern classification and scene analysis 2nd edition," in *A Wiley-Interscience publication*, p. 15, 2000.

[25] P. Sharma, "Comprehensive guide to build a recommendation engine from scratch (in python)." Availble at https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/ (2022/06/20).

[26] OldMonk, "Kaggle: Amazon electronics data." Available at https://www.kaggle.com/datasets/saurabhbagchi/amazon-electronics-data (2022/04/30).

[27] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," pp. 188–197, 01 2019.

[28] J. Ni, J. Li, and J. McAuley, "Amazon review data (2018)." Available at https://nijianmo.github.io/amazon/index.html (2022/04/30).

[29] I. Intelligence, "Worldwide ecommerce sales set to top $5 trillion for first time in 2022." Available at https://www.insiderintelligence.com/insights/worldwide-ecommerce-sales-to-top-7-trillion (2022/06/30).

[30] Instagram, "How to use live shopping to reach customers." Available at https://business.instagram.com/blog/instagram-live-shopping-reach-customers (2022/06/30).