Interpretable Recommender Systems

Practical Seminar

Alexander Rothmaier

2295961

At the Department of Economics and Management
Institute of Information Systems and Marketing (IISM)
Information & Market Engineering

Reviewer:	Prof. Dr. rer. pol. Christof Weinhardt
Advisor:	M. Sc. Alexander Grote

1th of February 2022

Contents

List of figures

1. Introduction

Every day our decisions are influenced by recommendation services of providers from different industries. For example, music or video streaming providers or even social media platforms or online shops nowadays use recommender systems which recommend items to their users. This simplifies our decisions but becomes dangerous when individuals rely on such recommendation services for their decisions and the models behind them become increasingly complex and non-transparent. Therefore, it is important to understand how a recommender system makes its decision. With the use of different machine learning methods, the state-of-the-art recommender systems reach higher accuracies, but therefore become more complex and the recommendation process becomes less comprehensible for the user. This situation, which is well known in the machine learning context, is called accuracy-interpretability trade-off.

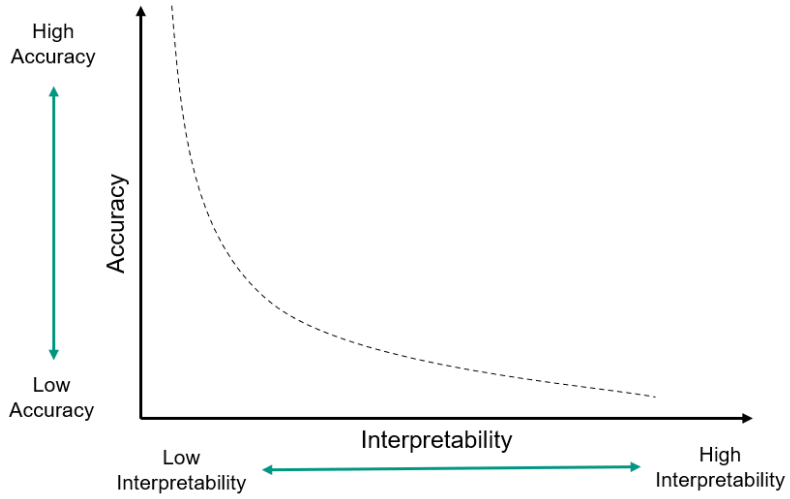

Figure 1: Accuracy-Interpretability Trade-Off

To counteract this and to increase trust and transparency in the recommendations made by the model, it is important to provide an additional explanation. Besides trust and transparency, the authors Tintarev and Masthoff distinguish between 7 different criteria for evaluating the explainability of recommender systems, shown in the table below.

Criterion	Definition
Transparency	Explain how the system works
Scrutability	Allow users to tell the system it is wrong
Trust	Increase users' confidence in the system
Effectiveness	Help users make good decisions
Persuasiveness	Convince users to try or buy
Effectiveness	Help users make decisions faster
Satisfaction	Increase the ease of use or enjoyment

Figure 2: Seven criteria for evaluating the explainability of recommender systems. Adapted from Table 15.1, Tintarev, N., & Masthoff, J. (2011).

2. Related Work

While building systems with high accuracies, which recommendations are additionally explainable, seems like the ultimate goal, there has been a lot of research within this area. Various authors propose different approaches to explain recommender systems. The explanation styles can be divided into two categories: embedded and post-hoc explanations.

Embedded explanations adjust the used recommendation model in such way, that an additional explanation is generated within the recommendation process and provided as output. This results in a white-box model. Therefore, these approaches depend on the underlying model and are tailored to them. Another way to provide explanations is to consider the recommender system as black-box model and to generate the explanations post-hoc. Both possibilities for explanation are shown schematically in the figure below.

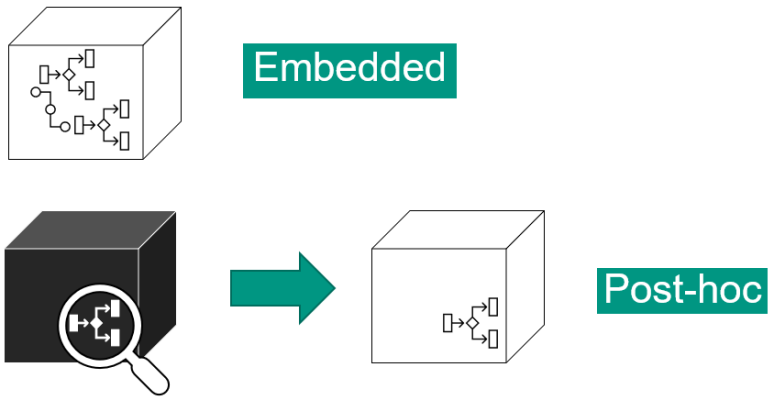

Figure 3: Embedded and post-hoc explanations

2.1. Embedded Explanations

As for embedded methods the explanation generation is integrated within the recommendation model, it makes sense to look at the main types of recommender systems and their possibilities for explanations separately.

2.1.1. Content-based

Content-based recommender systems are modelling user and item profiles with various available content information, such as the price, colour, brand of the goods in e-commerce, or the genre, director, duration of the movies in review systems [1]. Because the item contents are easily understandable to users, it is usually intuitive to explain the users why an item is recommended. For example, one straightforward way is to represent the decision driving content information within the recommendation process to the user. A popular approach combining different content information to create explainable recommendations is the "*Tagsplanation*"-approach proposed by John Riedl et. al (2009) [2]. Tags are used to characterize items in a high dimensional semantic space and to

explain recommendations, due to representing them to the user. The "*Tagsplanation*"-approach was presented for a movie recommendation system, generating tags from different sources. The tags were mined from textual reviews of users, plot descriptions of movies and other available metadata like actors, genres, directors, etc. Combining more content information about the items, the recommendations get better as well as the explanations do. The explanation is provided to the user by displaying the relevant tags for the current recommendation alongside with the relevance of the tag for the movie and the user's preference for that tag.

Figure 4: Exemplary "Tagsplanation" interface. Adapted from John Riedl et. Al (2009) [2]

However, collecting content information in different application domains is time-consuming and not always possible or sometimes it is very expensive to collect data [3].

2.1.2. Collaborative Filtering

Collaborative Filtering (CF) approaches tempt to avoid the problem of collecting content data, by leveraging the wisdom of the crowd [4]. User- and item-based CF are somewhat explainable due to the philosophy of their algorithm design. For example, the items recommended by user-based CF can be explained as "users that are similar to you loved this item", while item-based CF can be explained as "the item is similar to your previously loved items" [5]. These kinds of explanations are simple to generate, by additionally providing information about the most similar items or users to the output. This information must be carried internally during the recommendation process.

Latent factor models like matrix factorization are especially successful at predicting ratings in recommender systems and reach high accuracies in their predictions. The inherent problem is that "latent factors" in latent factor models do not possess intuitive meanings, which makes it difficult to understand why an item got good predictions or why it got recommended out of other candidates. To explain these "latent factors" or to at least prefer recommendations with higher explainability in the output, there are different approaches.

Zhang et. al (2018) propose the "*Explicit Factor Model*" as approach for explaining latent factor model's recommendations by aligning the latent dimensions with explicit features. By extracting explicit user opinions about various aspects of a product from the reviews, it

is possible to learn more details about what aspects a user cares, which further sheds light on the possibility to make explainable recommendations [6]. The authors first extract explicit product features and user opinions by phrase-level sentiment analysis on user reviews, then generate both recommendations and disrecommendations according to the specific product features to the user's interests and the hidden features learned. Explanations of the following type are generated: "You might be interested in [feature], on which this product performs well/poorly."

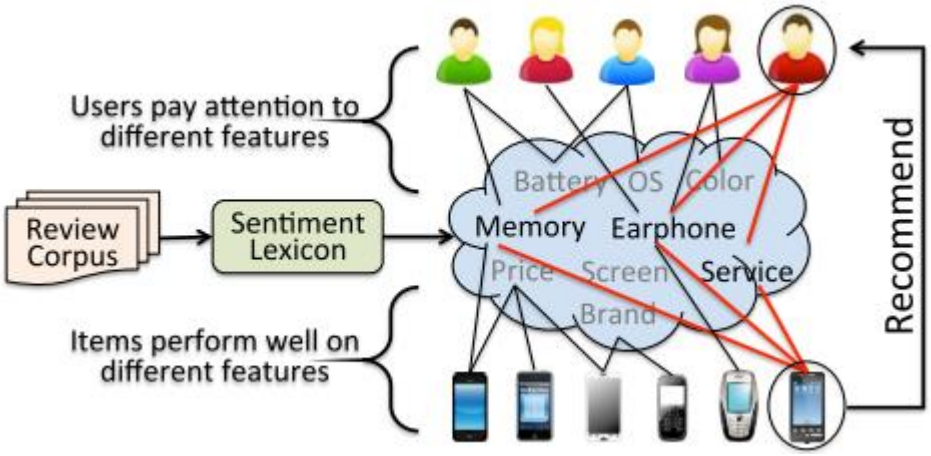

Figure 5: Schematic recommendation process, based on semtiment features which are mined from the text corpus. According to figure 1 in Zhang et. al (2018) [6]

Another approach which is additionally addressing the accuracy-interpretability trade-off is proposed by Abdollahi and Nasraoui [7]. The authors applied "*Constrained Matrix Factorization*" when training matrix factorization models by adding an "explainability score" as an additional input alongside the ratings given by the users. The "explainability score" is calculated mathematically and is higher if the current user has rated many similar items, or if similar users have rated the current item (i.e., "Similar to items you liked..." or "Users similar to you liked..."). The resulting recommender system gave higher rankings for items deemed explainable than what they would have had in a regular matrix-factorization-based recommender system. Although this approach yields recommendations which are more comprehensible and could be explained better, it does not generate explanations as output alongside the recommendations and is therefore not a type of embedded explanation.

2.2. Post-hoc Explanations

Post-hoc explanations are generated after the recommendation was made. By creating a separate component (explainer) an explanation given by a black-box (i.e., uninterpretable) recommender system can be generated. The key advantage of post-hoc explainability over other approaches is that it's model-agnostic: the same explainer can generate explanations for all kinds of models and requires no domain-specific additional data to work [8]. In the literature two common methods of post-hoc explanations have emerged.

2.2.1. Influence Analysis

An intuitive way to measure which system interactions (e.g., prior ratings given to movies which have already been watched) affected the recommendation the most is the *influence analysis*. The main idea of this approach is to successively omit the previous interactions with the system and to re-calculate the recommendation scores of all other items. The absolute difference in the recommendation scores for all unseen items is then added up. This is done for every previous interaction to find out, which interaction influenced the system's recommendation the most. Since there is a re-calculation of all other scores for every previous interaction this method is very computationally intensive. In response to this problem, the authors Weiyu Cheng et. al proposed a method called "*Fast Influence Analysis*" in their paper [9]. The idea is to reduce the computational costs, by not re-calculating the whole model, but only the parts of the system which are affected by the omission of the interaction. In order to reduce computational costs even further, statistical influence functions are used to estimate the differences. The method was proposed for latent factor models, specifically for matrix factorization [9] [10].

2.2.2. Association Rules

The authors Peake and Wang (2020) [11] proposed another post-hoc approach by extracting logical association rules from the recommendation model's input and output. The extracted rules were of the form {X => Y}, where X is an item, the user has rated before, and Y is recommended. The association rules can be expressed as "because you liked X, we recommend Y". The rules are mined using the output of the black-box model. The globally learned association rules are filtered for each user. Restricting the antecedents to be part of the items interacted with and the consequents being part of the items recommended by the black-box model. In their paper, the researchers limited the size of the association rules to just two items (X and Y in the above example) – experimenting with rules consisting of more items is an important consideration for further study. While a matrix factorization model was used in the research, the recommendation model is model-agnostic, meaning that it can generate explanations for any model [12]. A more detailed description as well as an approach for the implementation of the method is provided in the 'Results' section.

2.3. Visualising Explanations

After the explanations have been generated, it is important to decide how to represent them to the users. As the best explanation is useless, if the user can't understand it, it is essential to have a comprehensible visual representation of the generated explanations. There are some studies dedicated to the discovery of interfaces for explanations, which are most liked by users of recommender systems [13], [14]. The design of the interface is dependent on the goals, followed by the explanations. E.g., if the goal is to build thrust in the system, the actual design of the representation can be different from an interface with the goal of scrutability or persuasiveness. Therefore, it is important to consider the

different dimensions of explanation goals (figure 2). As there are many different types of explanation styles for recommender systems, there are also various approaches to visualize them. The following subsections provide an overview, just to cover some of them.

Collaborative Filtering

Herlocker et al. [15] evaluated several approaches to explanation in the collaborative movie recommender *MovieLens* in terms of their effects on user acceptance of the system's recommendations especially for recommendations generated by collaborative filtering approaches.

The most convincing explanation of why a movie was recommended was one in which users were shown a histogram of the ratings of the same movie by similar users. Moreover, grouping together of good ratings (4 or 5) and bad ratings (1 or 2) and separation of ambivalent ratings (3) was found to increase the effectiveness of the histogram approach [16]. Another important finding was that some of the explanations evaluated had a negative impact on acceptance, the goal of explanation in this instance, showing that no explanation may be better than one that is poorly designed.

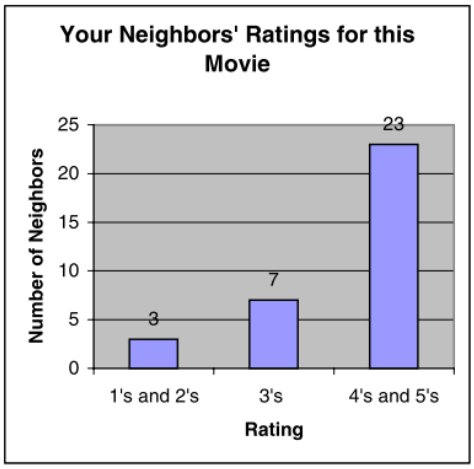

Figure 6: Grouped histogram showing neighbor's ratings [15]

Content-based

Content-based recommender systems as the name suggests, rely on the content of the items, which can be highly diverse. The relevant content of e.g., movies (title, director, genre, actors, duration, plot, …) can be very different from the relevant content of cameras e.g. (price, colour, megapixel, …).

Recommendations can be explained by displaying the performance of the recommended item among specific features and the user's attitude towards them. For example, a camera can be recommended to the user, because it is cheap and affordable and this is an important feature for the customer, as he previously bought low-priced and inexpensive products with a good price-performance ratio. Analogue in the movie example,

recommendations can be explained by giving an overview over the genres of a movie and the user's attitude towards them.

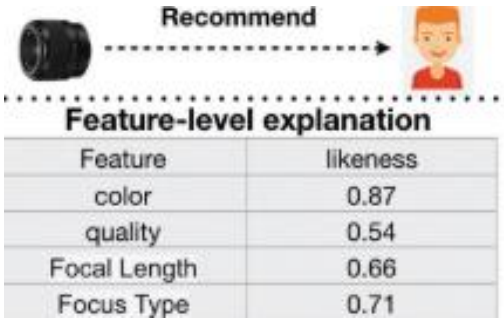

Figure 7: Recommendation explanation based on item features [5]

Usually, relevant-item explanations are more intuitive for users to understand because users are familiar with the items they interacted before. As a result, these items can serve as credible explanations for users, as shown in the figure below.

Figure 8: Relevant item explanation interface according to [5]

Visualising Association Rules

There is research on how best to visualize association rules in general [17], [18]. Fernandez-Basso et al. [17] compared different tools and styles for visualization according to their advantages and disadvantages. The basic representation as a table, is useful to observe a fragment of the complete set of rules, but not for a complete view for all discovered rules. On the contrary, the graph-based visualizations can be used for a more complex and complete representation of rules.

3. Methodology

3.1. Literature Search

To gain a rough overview over existing types of recommender systems and different types of explanation styles as well as dimensions of explainability, I used the work of Francesko Ricci et. al (2010) 'Recommender Systems Handbook' as a starting point [12]. Diving deeper into the different possibilities of explaining recommendations, I came across the work of Zhang et. al (2020) 'Explainable Recommendation: A Survey and New Perspectives' [3]. This paper provides an extensive literature review with a collection of references for further study of different explanation styles.

3.2. Python Implementation

Since Python is the leading programming language in terms of data science and machine learning, the decision was made to use Python and its various open-source libraries for the practical part of the seminar. To keep the overhead as low as possible, I decided to build the frontend with *Streamlit*, which is an open-source python package.

The *MovieLens* datasets, first released in 1998, describes people's expressed preferences for movies. These preferences take the form of <user, item, rating, timestamp> tuples, each the result of a person expressing a preference (a 0–5 star rating) for a movie at a particular time [19]. There are different datasets available, for implementation and training purposes I chose the smallest dataset, which contains roughly 100k ratings. The presented algorithms and dashboards in the section 'Results' are all applicable to the bigger 2m dataset, resulting in longer computation times. As the same dataset is used, the preprocessing is almost the same for all implementations: removing duplicates and unimportant features like 'timestamp' and creating the user-item interaction matrix as a pivot-table.

4. Results

According to the different major types of recommender systems and different explanation styles, I decided to implement four different approaches to cover most of them exemplary. For each approach an own dashboard was built, which will be discussed in the following part.

4.1. Embedded Explanations

4.1.1. Content-based using genres

Transferring the idea of displaying the decision-driving content information of recommended items to the movie recommendation setting, the movie genres serve as our feature dimensions. The movies within the *MovieLens* dataset can have 19 different genres. To extract the user's preference towards the different genres g, the user's ratings

for the previous seen movies s must be extracted. Furthermore, a one-hot encoding for the movies and their genres is needed. By transposing the one-hot encoded matrix and multiplying it with the ratings matrix as shown below, the result is a $(g\ x\ 1)$ matrix with 'preference' scores for each genre.

T

Title\Genre	Action	Comedy	Children
Toy Story	1	1	1
Shrek	0	1	1
Hangover	1	1	0
Ted	0	1	0

*

Title	Rating
Toy Story	5
Shrek	3
Hangover	5
Ted	1

=

Title	Preference
Action	10
Comedy	14
Children	8

Figure 9: Multiplication of the seen movies with their one-hot encoded genres and the user's ratings for the movies resulting in a preference score for each genre.

Now that the user's preferences towards the genres are known, scores for all missing ratings can be calculated, by multiplying the preference matrix with a matrix containing the one-hot encoding of the genres for all unseen movies. This results in a $(u\ x\ 1)$ matrix, where u is the number of unseen movies. This matrix can then be sorted descending by the scores and the top-n movies can be recommended alongside with their genres and the user's preferences. Resulting in explainable recommendations as output.

Title\Genre	Action	Comedy	Children
Terminator	1	0	0
Hangover 2	1	1	0
Avengers	1	0	0
Spiderman	1	1	1
Karate Kid	1	0	1

*

Title	Preference
Action	10
Comedy	14
Children	8

=

Title	Score
Terminator	10
Hangover 2	24
Avengers	10
Spiderman	32
Karate Kid	18

Figure 10: Multiplication of the unseen movies with their one-hot encoded genres and the user's genre preferences resulting in a recommendation score for each unseen movie.

As the author Herlocker evaluated different styles of explanation and found out that histograms or bar charts performed best in his studies, the user profile showing his preferences can intuitively be displayed as horizontal bar chart [15].

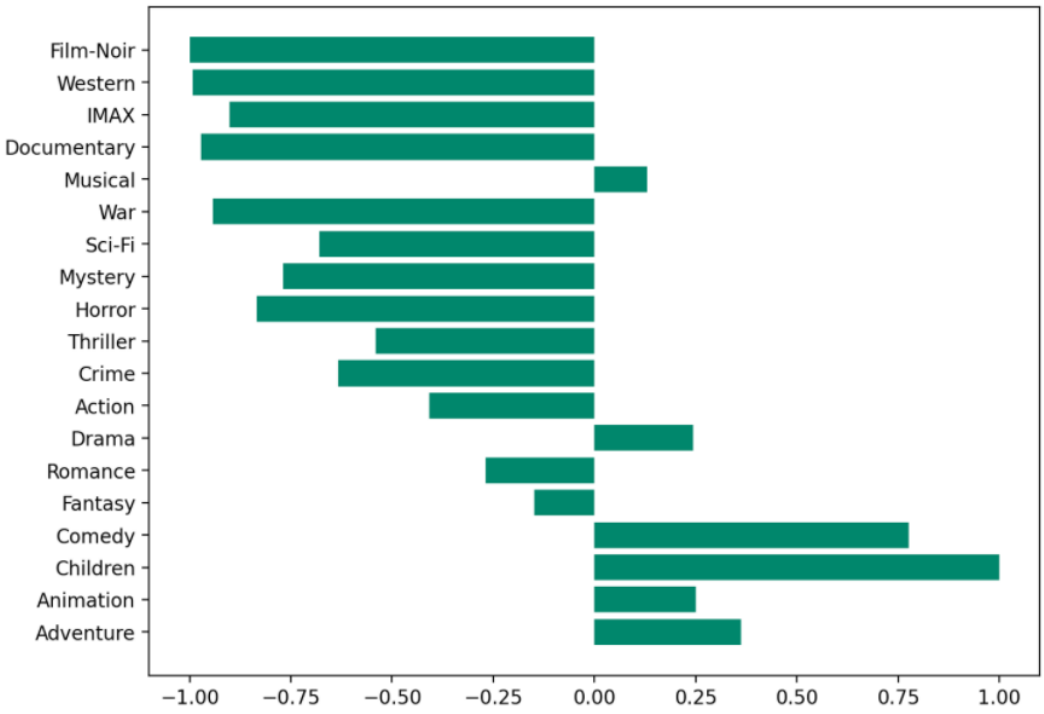

Figure 11: Horizontal bar chart showing genre preferences of selected user. The degree of liking or aversion towards the different genres is displayed as the magnitude of the bars.

For a clearer representation the preferences can be standardized between -1 and 1. The user can see how the system perceived his preferences based on his previous ratings. This might fit the user's personal perception or not, in any case it increases the *transparency* and *scrutability* of the system. The recommendations based on the preferences are traceable and the user is able to tell that the system may be wrong, because for example he does not even like children or comedy movies. Such a representation is complemented by the recommended movies and their genres. For example, it is now obvious, that the recommendation of 'Aladdin' as a children movie was made because the system's perception based on the previous given ratings of the user is that he likes children movies.

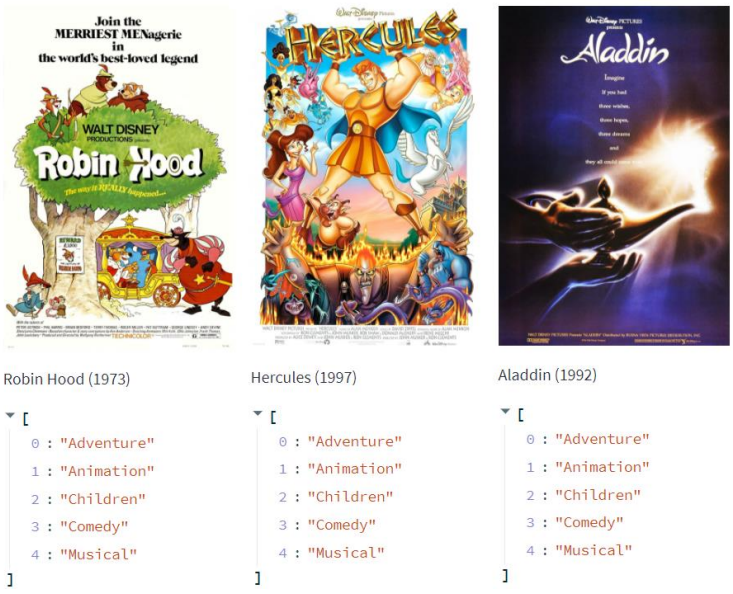

Figure 12: Recommended movies with their genres based on which they are suggested by the system

4.1.2. Content-based using plot

To cope with the conceptual problem of the 'genre approach' (discussed in Section 5) and to get a more accurate vectorial representation of the movies another approach is to use the textual description of the plot for each movie. Unfortunately, the original '100k' *MovieLens* dataset does not provide metadata information about the movie plot. Therefore, the movie plots had to be extracted from an older *MovieLens* dataset and had to be matched with the present dataset. Using the IMDB-Ids of the movies, a mapping from the movie titles to the plots could be created and the movie plot could be provided as additional feature to describe the movies. Only 0.02% of the movies could not be matched with a textual plot description. After the preprocessing was done, a *tfidf-vectorizer* of the scikit-learn package was used to create vector embeddings for each movie. Removing English stop-words and restricting the occurrence of words in different plot descriptions between a minimum of two and a maximum of 70% of all plots, the vectorizer leveraged over 16.000 words. The result was a $(9737x16118)$ matrix, where the rows are vectorial representations of the textual plot descriptions. In the *tfidf-approach*, scores representing the relevance of each word for each document (plot description) are calculated using the following formula:

$$tfidf(i,j) = tf_{i,j} * \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j} = number\ of\ occurences\ of\ item\ i\ in\ document\ j\ (term\ frequency)$
$df_i = number\ of\ documents\ j\ containing\ the\ term\ i\ (document\ frequency)$
$N = total\ number\ of\ documents\ within\ the\ corpus$

The representation of each movie is a high-dimensional numerical vector with the *tfidf-scores* as entries. The pairwise cosine-similarity for two numerical $nx1$ vectors is then calculated the following way:

$$cos\_sim(a,b) = \frac{a * b}{\|a\| * \|b\|} = \frac{\sum_{i=1}^{n} a_i * b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} * \sqrt{\sum_{i=1}^{n} b_i^2}}$$

By calculating the cosine-similarity between the 9737 row vectors, this yields a $(9737x9737)$ matrix with the pairwise similarities between the movies as entries. The usage of the cosine-similarity was chosen arbitrarily, and it is to say, that any other metric can be chosen to calculate the distance between the item-vectors.

As the features are too high-dimensional (> 16.000) a visual representation of the feature characteristics to the user is impossible. Therefore, the previously undertaken approach is not applicable in this use-case.

Since the top-m similar movies given to a specific movie x can be effortless calculated with the usage of the previous similarity matrix, I developed a new approach for explainable recommendations. A 'vote score' for each of the m most similar movies is calculated, by multiplying the cosine-similarity with the rating, the user gave to the movie x. As we have the problem, that the users' rating behavior is heterogenous, the ratings must be standardized between -1 and 1 before. Otherwise, users with ratings between 0-3 have a different impact than users which tend to rate higher between 4-5.

$$\hat{r}(x) = -1 + \frac{2 * (r(x) - \min_R r)}{\max_R r - \min_R r}$$

$$vote\_score(x,y) = \cos\_sim(x,y) * \hat{r}(x)$$

This is done for all movie ratings *R* which the user rated previously. The 'vote scores' for each movie are then added up, resulting in a 'recommendation score' for the unseen movie y.

$$rec\_score(y) = \sum_{x \in R} vote\_score(x,y)$$

This approach yields a list of recommended movies, alongside with their 'recommendation scores' and 'root movies', as well as the ratings, the user gave to them previously.

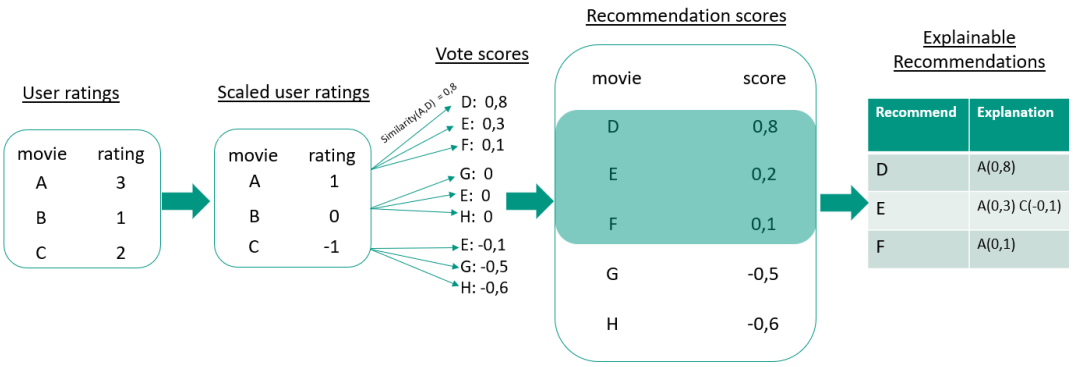

Figure 13: Schematic process of generating explainable recommendations with vote scores

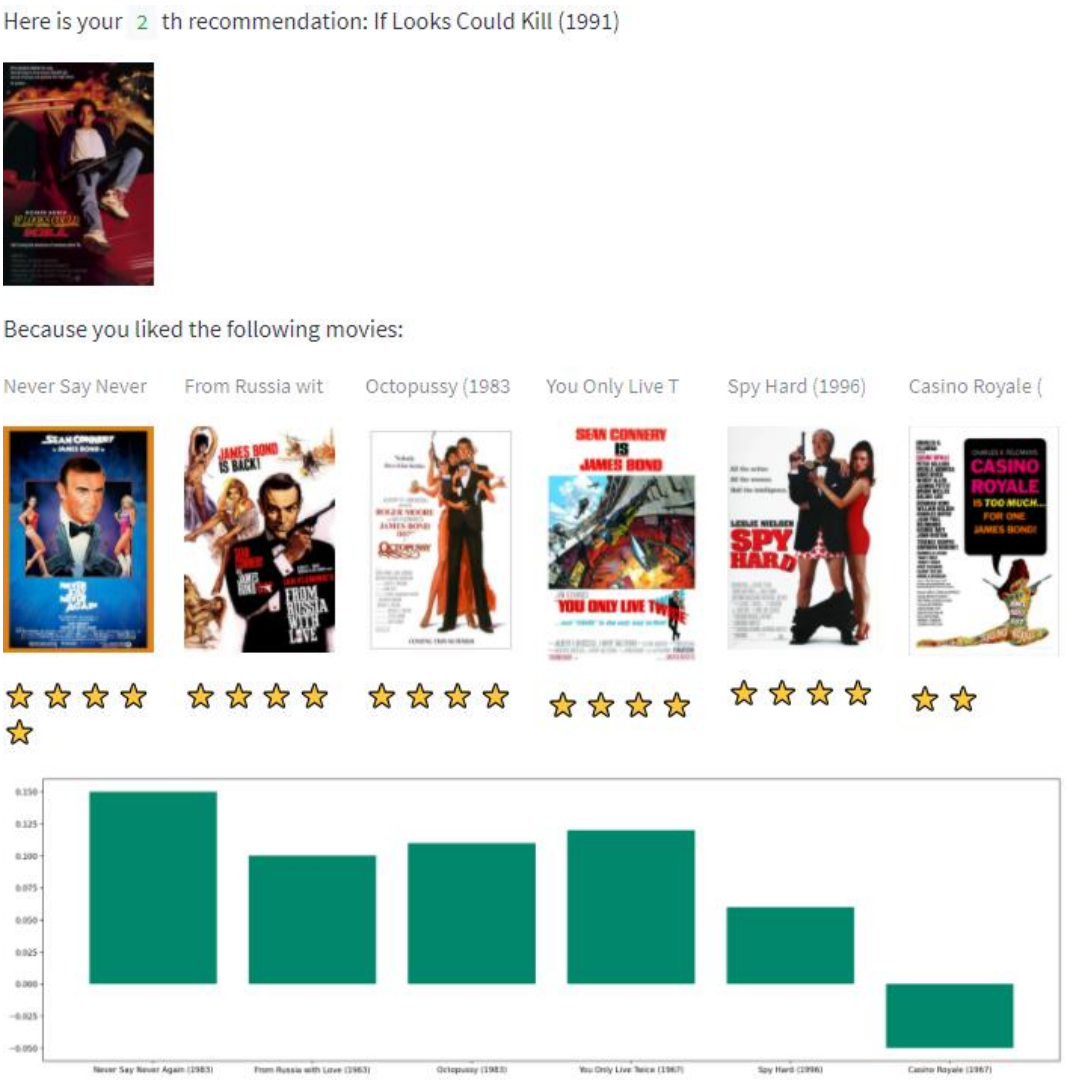

Figure 14: Dashboard showing the influences of previous ratings on the recommendation of the system

The user is able to track which of his previous ratings affected the recommendation in what manner. It is also possible to show the influences of each rating to the user as a bar chart (Figure 14). The magnitude of the bars is the percentage of influence of the movie;

therefore, the sum is 100 percent. Providing a representation like shown below to the user, results in a higher transparency and satisfaction. The user can see how the system works and which of his previous ratings affected the decision the most. As the recommendation process gets more transparent, the user's trust in the system is increased as well. However, not all criteria are increased with this type of explanation. For example, the decision making is not faster, and the user is not more capable of telling the system is wrong.

4.1.3. Collaborative Filtering (User-User)

In the collaborative filtering approach of recommender systems, the single source of information is the user-item interaction matrix. After preprocessing the data of the *MovieLens* dataset and fitting it into a $(610x9737)$ matrix where the rows represent the users, and the columns represent the movies, the collaborative filtering can be applied. [1]

Calculating the cosine-similarity between the rows, yields a $(610x610)$ matrix, containing the pairwise similarities between the users. Again, other metrics are possible, and the cosine-similarity is chosen arbitrarily. With the usage of the similarity matrix the k-nearest neighbors (most similar users) can be calculated. Whilst excluding movies which have already been watched and rated by the regarded user U, recommendations can be made. Again, I chose the 'vote score' approach. Therefore, the ratings of the k-nearest neighbors must be standardized between -1 and 1 to account for different voting behavior. Afterwards, movies already watched by the k-nearest neighbors are stored in a unique list. As these are candidates for recommendations, it is important, that there are no movies in the list, which the target user has already seen. The neighbors' ratings to these movies are multiplied with their similarity to the regarded user U and then added up as 'vote score'. The 'root users' which gave the ratings are then stored alongside with their rating. After sorting the list descending by the 'vote score' the top-n movies can be output as recommendations in combination with their 'root users' and their 'root-ratings'.

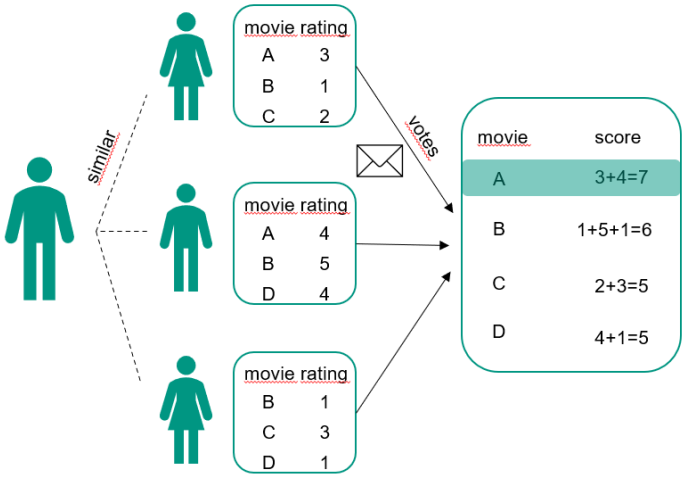

Figure 15: Schematic process of generating explainable recommendations with the votes of neighbors

[1] By transposing the matrix, the procedure can be applied analogously for item-based collaborative filtering.

As it is not in the interest of the user to find out about which users, he is similar to an anonymized representation of the results is needed. Also, in regard to privacy aspects. The results can be visualized as a table or histogram, showing how many stars each of the k-nearest neighbors gave to the recommended movie. It can be also thought of a grouped histogram where 1-2 stars, 3 stars, and 4-5 stars are grouped together for a better overview (Figure 17).

	stars	#votes of 10 neighbors
0	☆	0
1	☆☆	0
2	☆☆☆	0
3	☆☆☆☆	1
4	☆☆☆☆☆	8

Figure 16: Table representation of neighbors' ratings for a recommended movie

I want to remark, that neighbors who have not watched the recommended movie do not appear in the representation. Therefore, the sum of the given votes is meaningless.

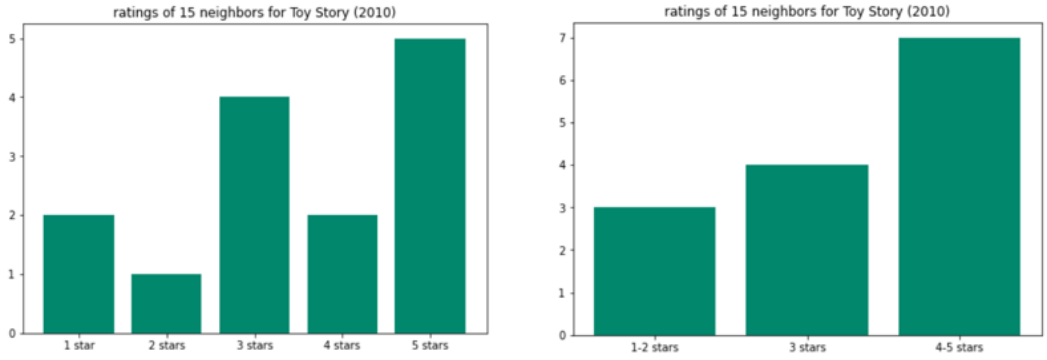

Figure 17: Histogram representation of neighbors' ratings for a recommended movie

4.2. Post-hoc Explanations

4.2.1. Association Rules

Starting with a black-box model using matrix factorization to generate recommendations, I followed the approach proposed by Peake and Wang, shown in the figure below [11].

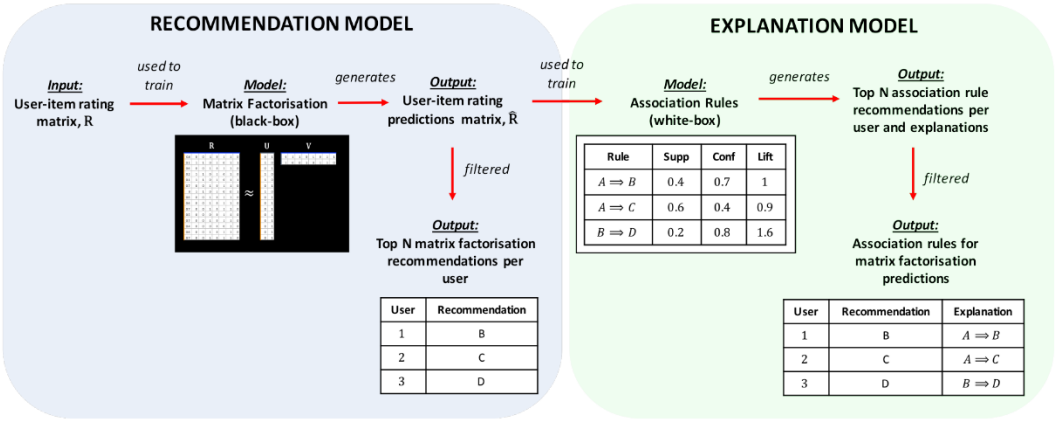

Figure 18: Schematic diagram of proposed approach for generating an explanation model by training association rules on the output of a black-box matrix factorization recommendation model [11]

After the regular preprocessing of the dataset (described in Section 3), I subtracted the row mean in each row of the user-item interaction matrix to center the ratings of each user around zero. Then I filled the missing NaN-values with zeros. Next, the regular matrix factorization method was used to predict the missing ratings within the sparse matrix. The matrix with its predicted ratings can then be stored as a file and loaded to save time and computational resources. Considering the movies which the user has already seen, the top-n unseen movies with the highest predictions get recommended.

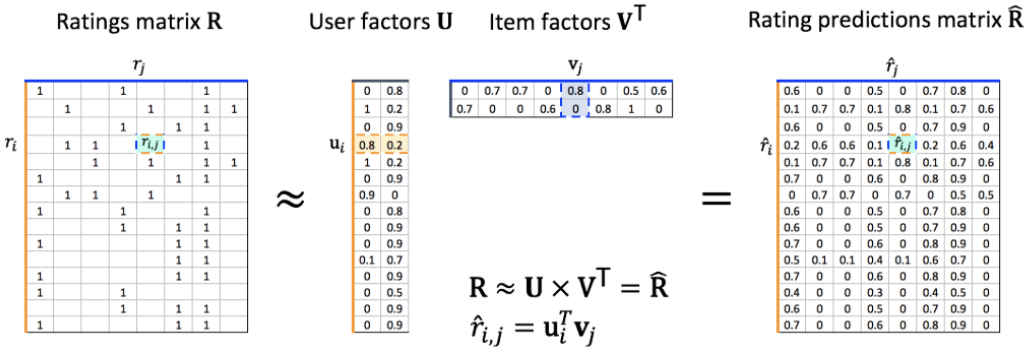

Figure 19: Matrix factorization is an example of a latent factor model, which detects a given number of factors, k (also known as rank), by compressing the user-item matrix R into two lower rank matrices. By calculating the dot product of U and $\boldsymbol{V^T}$, the missing ratings are predicted within the matrix $\boldsymbol{\hat{R}}$. [11]

The output of the top-n predicted movies for each user was then stored in a list. This list was preprocessed with a *Transaction Encoder* and the *apriori algorithm* to filter out the most frequent item sets [20]. After that, the association rules can be mined, restricting the rules to the type X => Y with only one antecedent and one consequent. Furthermore, I

filtered out the rules with a *confidence* greater than 0.3 and sorted them descending by the lift. With this set of global association rules, the personalized rules for each user can be generated. For this purpose, lists of the predicted unseen and seen movies of a user are necessary. The global association rules can then be limited to those, where the antecedent is part of the seen movies, and the consequent is part of the predicted unseen movies. These rules are the final output, as the consequents are the predicted movies, which the user has not already seen, and the antecedent explains based on which already seen movie the recommendation was made. Resulting in a transparent white-box model, the predictions of the matrix factorization were left untouched, so there is no loss of accuracy due to the additional explainability of the recommendations.

The association rules as output can be visualized as a table or as a 2D network graph, as shown in the figures below. For most purposes the table representation should be sufficient and clear to understand, as the rules only possess one antecedent and one consequent.

	antecedents_t	consequents_t	support	lift
0	Notting Hill (1999)	Dirty Dancing (1987)	0.0033	203.0000
1	Star Wars: Episode IV - A New…	Tin Cup (1996)	0.0033	203.0000
2	Harry Potter and the Sorcere…	Harry Potter and the Goblet …	0.0033	152.2500
3	Harry Potter and the Prisone…	Toy Story 3 (2010)	0.0033	152.2500
4	Silence of the Lambs, The (1…	American President, The (19…	0.0033	121.8000
5	Beauty and the Beast (1991)	Fantasia (1940)	0.0033	121.8000
6	Beauty and the Beast (1991)	Snow White and the Seven D…	0.0033	121.8000
7	Star Wars: Episode IV - A New…	Father of the Bride Part II (19…	0.0033	121.8000
8	Silence of the Lambs, The (1…	Addams Family Values (1993)	0.0033	101.5000
9	Star Wars: Episode V - The E…	Big (1988)	0.0033	101.5000

Figure 20: Table representation of association rules for explaining movie recommendations

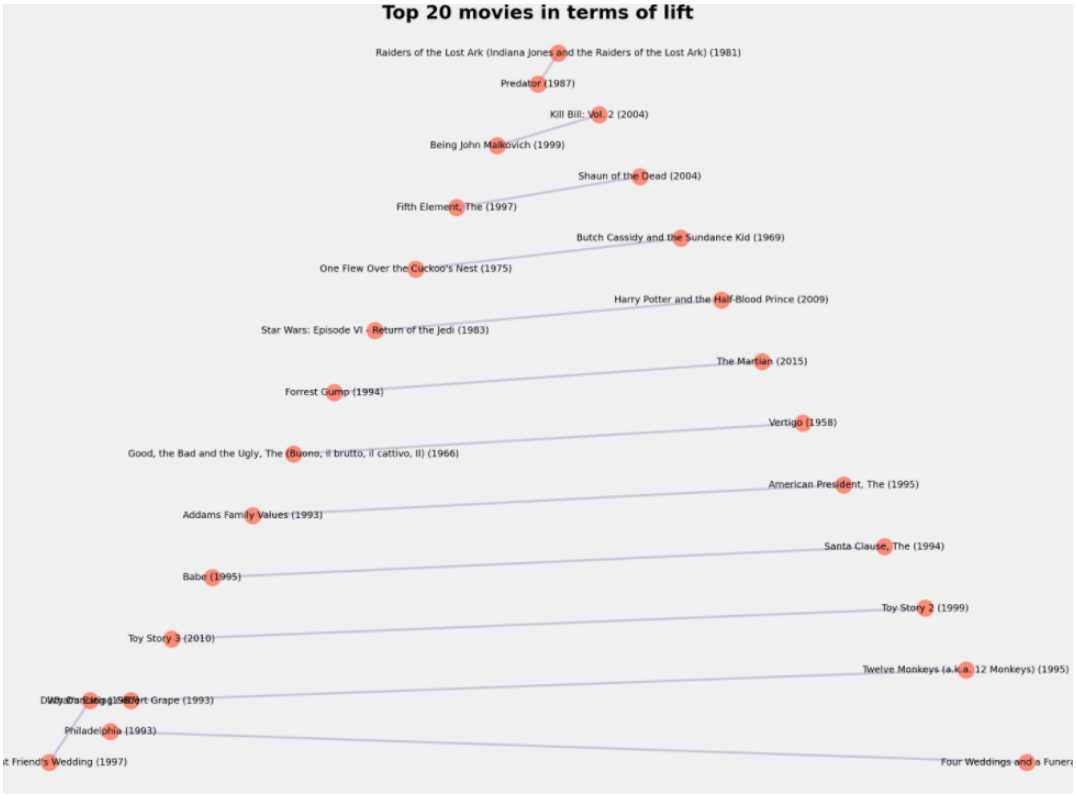

Figure 21: 2D network graph representation of association rules for explaining movie recommendations

4.2.2. Influence Analysis

An intuitive way to track the size of influence of previous ratings is the influence analysis. As described in the previous chapter, the goal is to successively omit the given ratings and to re-calculate the scores. Prerequisite is the presence of a score system for the unseen movies. This can be achieved in different ways for different recommender systems. For example, the score system in the *content-based approach using categories* is the multiplication of the genre preferences with the one-hot encoded genres of the unseen movies, resulting in a single score for each movie. In the *content-based approach using textual plot descriptions*, the scores are the results of the added-up votes, which are the product of the item similarity and the given rating. Using matrix factorization as a representative collaborative filtering method, the predicted ratings for unseen movies are scores in that sense as well. The model-agnostic method of the influence analysis works the same way for every of these models with a scoring system: by leaving out one rating at a time, the scores can be re-calculated, and the influence of the rating can therefore be measured (Figure 22). As re-calculations are needed for every previous given rating, this method is very computational expensive. Like previously described, there are approaches where there is only an estimation done and no complete re-calculation [11]. To try the influence analysis without relying on estimations, I decided to implement it within the recommender system, with the fastest re-calculation speed. Therefore, I chose the content-based recommender system using genres.

The calculations of the genre preferences, and the score calculations for the unseen movies were left untouched. I just added an additional method, which successively leaves one

rating out, re-calculates the scores and adds up the differences. It is possible to choose the absolute differences and add them up, to see which items have the biggest influences, whether positive or negative. Or it is possible to add up the actual differences to see, which items influenced the system's recommendation in a positive manner and which in a negative manner. The calculated differences are the influences, which can be output in the dashboard. As some users have more than hundred ratings, showing influences for every previous rating can be overwhelming for the user. Therefore, it could be helpful to only show the three most positive and negative influences.

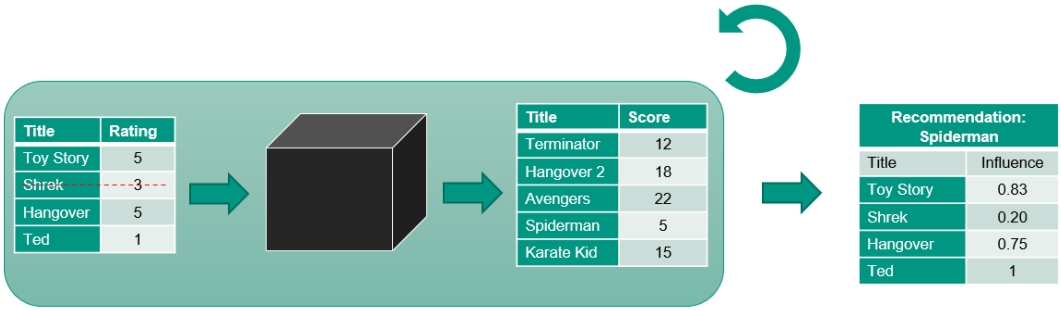

Figure 22: Schematic process of generating post-hoc explanations with influence analysis

5. Discussion

As the explainer and recommender components are not decoupled in embedded explained systems, they tend to naturally have good explainability. In terms of accuracy, it is unclear if embedding explanations to the model generally increases or decreases it's accuracy, as both positive and negative results have been reported [7], [8]. As post-hoc explanations leave the black-box recommendation systems untouched, the model's accuracy is not affected.

Limitations of the MovieLens dataset

The *MovieLens* datasets include data only from users with at least 20 ratings, as a result they are inherently biased towards frequent users [19]. Therefore, the users who are less interested in rating movies, were unable to find enough rateable content, or did not enjoy their initial experience in the system are not included in the datasets. It is possible that these users are fundamentally different from the users in the datasets.

Data Privacy

Regarding privacy aspects, it must be paid attention to what information about similar users is shown in the collaborative filtering approach. Relevant-user explanations, however, could be less convincing because the target user may know nothing about other "similar" users at all, which may decrease the trustworthiness of the explanations [15]. Some users might find it helpful to know more about the 'neighbours' based on which ratings a movie gets recommended. For example, a user could possibly have the opinion that a person who dislikes 'Star Wars' has a bad movie taste, and he is not willing to

receive recommendations based on such a person's opinion. Therefore, it would be an upgrade to the representation if some basic information about the 'neighbours' would be displayed to the user. This could be the favourite three movies of the users or some demographic information, such as age, gender, etc. Of course, taking care of not violating any privacy aspects.

Limitations of the content-based approach using genres

As the score of a movie is calculated by multiplying the quantitative genre preferences of a user with the one-hot encoding of the movie's genres, movies with many genres get overestimated. Because the genres preferences are non-negative values, movies with many genres have an advantage over those with less genres. As a result, it is observable that the movie 'Rubber (2010)' with its 10 genres is recommended exceptionally often. A potential solution to this problem could be a regularization of the scores, by dividing the movie scores with the number of genres. But this would in fact penalize movies with few categories. Possible approaches like a logarithmic penalty term are conceivable.

Moreover, using the 19 movie genres in the *MovieLens* dataset as feature dimensions seems still extendable, as the features are only binary. A movie can only possess a genre or not, but there is no quantitative metric about how strong the magnitude of the feature is, like it is the case for item features like price, length, durability etc. For example, the movie 'Hercules' is assigned to the genre 'musical' but is rather only a children movie when compared with other musicals like 'La La Land'. To conclude, the presented approach of content-based recommendations and explanations based on the 19 categories seems underperforming, when compared to higher-dimensional feature approaches.

Limitations of content-based approach using movie plots

As some of the recommendations seem very accurate and well fitting, others seem to be random. While retracing the output, some recommended movies seem to be completely unrelated with their 'root movies'. For example, the movie 'Hachiko: A dog's tale' gets recommended based on high ratings to multiple James Bond movies. Whilst looking at the movie plots of them, it is recognizable that the main protagonist of the movie is a professor called 'Bond'. As the word 'Bond' seldom appears in other texts, but frequently appears in the plots of 'Hachiko: A dog's tale' and the James Bond movies, it is characteristic for them. Therefore, the vector representations of the movies are very similar, even though the movie 'Hachiko: A dog's tale' is associated with completely different genres, narrates a different storyline, and has not very much in common with the James Bond movies. This is an inherent problem of the tfidf-approach. A possible solution to the problem could also be the extension by more descriptive features. For example, if the genres, plot, director, actors, and title would have been combined and taken into consideration, the recommender system would have recognized that the movies are semantically far from each other and would not have recommended it.

6. Conclusion

Addressing the accuracy-interpretability trade-off it is noticeable, that simpler recommender systems are easier to be interpreted. For example, the recommendations of systems using a content-based approach (genres or plot) are intuitively explainable, as the internally created user-preference-profile as well as the item features can be represented to the user without much effort. Equally simple collaborative filtering approaches, like the nearest neighbor approach e.g., can intuitively be visualized by representing the neighbor's opinions.

As models get more complex in order to increase accuracy, their interpretability decreases at the same time, as it is the case for latent factor models (e.g., matrix factorization). As the latent factors are inherently uninterpretable, it is not possible to simply represent the internal decision drivers of the recommender system to the user. Approaches like the "*Constrained Matrix Factorization (CMF)"* or "*Explicit Factor Model (EFM")* are conceivable. Although they do not seem applicable as the *CMF* method only focuses on outputting more interpretable recommendations but does not create explanations. The *EMF* method relies on textual user reviews as additional data sources. The only practical applicable options for explaining latent factor models (e.g., matrix factorization) seem to be model-agnostic explanations which are post-hoc created. Since these methods leave the black-box model and the recommendations untouched, they seem to be an adequate response to the accuracy-interpretability tradeoff. As these methods can get computational expensive (e.g. influence analysis) there is research about improving the computational performance of them, using estimation functions [9], [10].

In summary, the post-hoc explanations are a good response to the accuracy-interpretability trade-off, as the explainer and recommender component are decoupled, and the recommendations are left untouched by the explanation. Therefore, high accuracies reached by complex models are retained and the recommendations are additionally explained. However, these explanations are often not that detailed or accurate as embedded explanations are. Finally, it depends on the goals followed by a recommender system, which type of explanation style is to choose. If the primary goal of a system is to reach a good explainability, it could be useful to stick with a simpler model and to use an embedded explanation style.

7. Declaration about the thesis

Ich versichere hiermit wahrheitsgemäß, die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des Karlsruher Instituts für Technologie (KIT) zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, den 05.01. 2022 _____

Alexander Rothmaier

References

[1] M. Balabanovi and Y. Shoham, “Content-based, collaborative recommendation,” *Commun. ACM*, vol. 40, no. 3, pp. 66--72, 1997.

[2] J. Vig, S. Sen, and J. Riedl, “Tagsplanations: Explaining recommendations using tags,” *Int. Conf. Intell. User Interfaces, Proc. IUI*, no. May 2014, pp. 47–56, 2009, doi: 10.1145/1502650.1502661.

[3] Y. Zhang and X. Chen, *Explainable Recommendation: A Survey and New Perspectives*. 2020.

[4] M. D. Ekstrand, “Collaborative Filtering Recommender Systems.” 2011.

[5] Y. Zhang, X. Chen, and B. -Delft, “Foundations and Trends ® in Information Retrieval Explainable Recommendation: A Survey and New Perspectives,” *Found. Trends Inf. Retr.*, vol. 14, no. 1, pp. 1–101, 2020, doi: 10.1561/1500000066.Yongfeng.

[6] Zhang, “Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis,” vol. 30, no. 1, pp. 1–6, 2018, doi: 10.1053/j.semtcvs.2018.01.002.

[7] B. Abdollahi and O. Nasraoui, “Using explainability for constrained matrix factorization,” *RecSys 2017 - Proc. 11th ACM Conf. Recomm. Syst.*, pp. 79–83, 2017, doi: 10.1145/3109859.3109913.

[8] D. Shmaryahu, G. Shani, and B. Shapira, “Post-hoc explanations for complex model recommendations using simple methods,” *CEUR Workshop Proc.*, vol. 2682, no. c, pp. 26–36, 2020.

[9] W. Cheng, Y. Shen, Y. Zhu, and L. Huang, “Explaining Latent Factor Models for Recommendation with Influence Functions,” 2018, doi: 10.1145/3292500.3330857.

[10] W. Cheng, L. Huang, Y. Shen, and Y. Zhu, “Incorporating interpretability into latent factor models via fast influence analysis,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 885–893, 2019, doi: 10.1145/3292500.3330857.

[11] W. Peake, “Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems,” *L@S 2020 - Proc. 7th ACM Conf. Learn. @ Scale*, pp. 321–324, 2020, doi: 10.1145/3386527.3406738.

[12] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, no. October. 2011.

[13] N. Tintarev and J. Masthoff, “A survey of explanations in recommender systems,” *Proc. - Int. Conf. Data Eng.*, pp. 801–810, 2007, doi: 10.1109/ICDEW.2007.4401070.

[14] P. Pu and L. Chen, “Trust building with explanation interfaces,” *Int. Conf. Intell. User Interfaces, Proc. IUI*, vol. 2006, pp. 93–100, 2006, doi: 10.1145/1111449.1111475.

[15] J. L. Herlocker, J. A. Konstan, and J. Riedl, “Explaining collaborative filtering

recommendations," *Proc. ACM Conf. Comput. Support. Coop. Work*, no. November 2015, pp. 241–250, 2000, doi: 10.1145/358916.358995.

[16] D. McSherry, "Explanation in recommender systems," *Artif. Intell. Rev.*, vol. 24, no. 2, pp. 179–197, 2005, doi: 10.1007/s10462-005-4612-x.

[17] C. Fernandez-Basso, M. Dolores Ruiz, M. Delgado, and M. J. Martin-Bautista, "A comparative analysis of tools for visualizing association rules: A proposal for visualising fuzzy association rules," *Proc. 11th Conf. Eur. Soc. Fuzzy Log. Technol. EUSFLAT 2019*, vol. 1, no. Eusflat, pp. 520–527, 2020, doi: 10.2991/eusflat-19.2019.72.

[18] A. Ceglar, J. Roddick, P. Calder, and C. Rainsford, "Visualising hierarchical associations," *Knowl. Inf. Syst.*, vol. 8, no. 3, pp. 257–275, 2005, doi: 10.1007/s10115-003-0139-0.

[19] F. M. Harper and J. A. Konstan, "The MovieLens Datasets," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2016, doi: 10.1145/2827872.

[20] M. Al-maolegi and B. Arkok, "An improved apriori algorithm for association rules," vol. 3, no. 1, pp. 21–29, 2014.