

Fairness, Unbias and User Privacy in Recommender Systems

Tzianara Polikseni

Supervisor: Symeonidis Panagiotis

University of the Aegean

Postgraduate Programme: Information and Communication Systems

September 2023

Contents

1	Introduction	3
2	Chapter 1 - Fairness and bias in recommender systems	4
2.1	Definition of a fair and unbiased recommender system	4
2.1.1	Taxonomy of fair recommender systems . .	5
2.1.2	Multisided fair recommender systems	5
2.1.3	Consumer fair recommender systems	6
2.1.4	Provider fair recommender systems	6
2.1.5	Fair recommender system for consumers and providers	6
2.2	Bias taxonomy	7
2.2.1	Popularity bias	7
2.2.2	Exposure bias	8
2.2.3	Position bias	9
2.2.4	Conformity bias	10
2.3	How to create a fair and unbiased recommender system	10
2.3.1		10
2.4	Categorizing recommender systems against popularity bias	42
3	Chapter 2 - Aspects of privacy in recommender systems	59
3.1	Risks of Privacy in Recommender Systems	60
3.2	Risks Caused by the Recommendation System .	62
3.3	Risks from conclusions drawn for new data	64
3.4	Risks Other Users Impose	67
3.5	Risks originating from external entities	69
3.6	Solutions on Privacy	71

3.7	System Design Architectures and Solutions	72
3.8	How to build a privacy-respecting recommender system	84
3.9	Categorization of privacy-respecting recommender systems	103
3.10	A new fair recommendation system	111
3.11	A user privacy-oriented recommendation system	112
	Contents	

1 Introduction

We are amidst the most exciting period from the moment the internet came into our lives. Today we can find anything on the web, ranging from information about public means of transport arrival and departure times to where and how anyone can buy a product of interest. However, there is a huge amount of information available, with people being unable to manipulate it and not knowing what to choose. At this point, Recommender Systems make their debut. Recommender systems have been incorporated in every website possible and based on the needs of consumers and distributors, they decide what to recommend to consumers. Let us imagine, for instance, that a consumer wants to buy a pair of shoes. They will connect to a website and start browsing at shoes. Unfortunately, sometimes a consumer has not decided what kind of shoes they want to buy. A recommender system can propose a solution to this problem. The recommender system, depending on the algorithm used, will make concrete recommendations to the consumer. But what are recommender systems and how exactly do they operate? This is a matter to be discussed on the first chapter of this paper. Issues of fairness on recommender systems will be considered on the second chapter. On the third chapter of this paper the topic of unbiased on recommender systems will be analyzed and on the fourth and last chapter of the paper the discussion will be centered around the interesting issue of user privacy on recommender systems.

2 Chapter 1 - Fairness and bias in recommender systems

2.1 Definition of a fair and unbiased recommender system

Recommender systems, search engines and other information systems filter many of our experiences when it comes to the information we share. However, they exhibit a series of problems about the way we can see the information world. Many of these problems come from failure to understand the social context of the design, the evaluation and execution of these information systems. As a result, problems that have not been diagnosed in those systems can reproduce unwanted consequences. As information systems continue to engage in different areas, it is vital that engineers understand how these systems interact with our society. In the past years, many researchers have published papers on the construction of machine learning models, which adapt the users' behaviour data, but when it comes to using these on real platforms, the recommender systems cope with bias problems, which can become a real challenge if someone tries to manage them and maybe even worsen the recommendations' effectiveness. A blind application of a recommendation generation model without bearing the underlying bias in mind could lead to distorted results. This is evident from the following factors:

Users' behaviour data, which lay the foundations for training recommendation generation models, are more useful for observation rather than experiments. The main reason is that a user behaves in a particular way and depending on the items they are exposed to, they misinterpret the observation data with the system's exposure mechanism. The blind use of the model with "biased" data could lead to a distorted image of the user's preferences.

Items are not equally represented in the data, meaning that some items are more popular than others and therefore receive more attention from the users. These popular items can have a bigger impact on the training model of the system, generating recommendations that have bias towards them. The same thing happens to users as well.

Feedback loop is a phenomenon in recommender systems – the system's exposure mechanism that generates recommendations defines the users' behaviour,

which come back as training data. Such a feedback loop not only creates bias, but, also, it makes them even more intense as time goes by, leading to Matthew’s accumulated wealth phenomenon.

There are many definitions for fair recommender systems. Their definition, though, presents challenges. In recommender systems, there is a population of vulnerable persons, called “protected group” and can be defined with a variable value on which some factor is based (i.e., sex, ethnicity or religion). A fair recommender system should provide the same results to the protected group as well as to the unprotected group.

2.1.1 Taxonomy of fair recommender systems

Algorithms should be carefully designed in order to satisfy the demands of recommender systems. This happens, because the idea of recommendation on recommender systems has been expanded to multiple stakeholders. For example, the lack of fairness in recommendation shouldn’t be examined only from one aspect, but from all aspects .

2.1.2 Multisided fair recommender systems

Recommendation procedures in multisided platforms (those which take into account every factor) could present problems regarding if recommendations are in fact fair. Particularly, there could be some criteria when it comes to this matter in different aspects of a transaction: it cannot empower only one side. A transaction has to give every participant the chance to be represented. There are three kinds of systems: the systems on the side of the consumers, the systems on the side of the providers and the systems on both sides.

2.1.3 Consumer fair recommender systems

A system that takes the side of the consumer into account is the system which calculates the effect of the recommendations on the protected groups of the consumers. For instance, a consumer wants to buy clothes from a clothing company and searches for clothes on the website of the company. There are no problems for the provider side, because the profit goes to the company. The problem that surfaces is on the consumer's side.

2.1.4 Provider fair recommender systems

A system that acknowledges only the side of the provider is a system that is fair for service providers. A good example of such a system is Kiva.org, which is an online microlending website. This website collects loan requests from contributors from all across the world, who lend small deposits of money to entrepreneurs in their local communities. The loans are funded free of interest from Kiva's members, mainly in the USA. Kiva doesn't offer any personalized recommendation service yet, but if it did, anyone could think that the organization's objective is to maintain a fair division of capital between all its collaborators.

The recommendation consumers are in fact donors, so they don't profit from the system, thus there are no obvious problems regarding fairness from the consumers' part. If the recommender system is fair only for providers, it can increase diversity in the market and it can stop monopolies. This kind of fairness might not be imposed by the law, but it is inherent in the model of a platform. There exist difficulties with systems taking the providers' side into account, which are not present on systems on the side of the consumers. Providers in recommender systems on the side of the provider are passive: they do not look for recommendations, but expect consumers to enter the system and ask for recommendations (Sonboli and Burke 2019).

2.1.5 Fair recommender system for consumers and providers

Finally, a platform that examines both sides (consumers and providers) may demand that the system be fair for both sides. This can happen in cases of mutual recommendations or in some field in which consumers as well as providers belong

to the protected groups. For example, a recommender system for a house rental company could manage user minorities similar to those suggested to white tenants. Simultaneously, the system could, also, wish to manage residence manager minorities as a protected group and to ensure that their residences be suggested to good tenants as it would for white residence managers.

2.2 Bias taxonomy

2.2.1 Popularity bias

Recently there has been an increasing interest in other sides of recommendations, such as unbiased and fairness. The imbalance problem of a class in machine learning is one of the main reasons for unfair ranking. For instance, it is widely known that face recognition systems entail racial bias, because many different races are not equally represented in the training data. Popularity bias in recommender systems is the equivalent of class imbalance in machine learning: popular items are frequently rated, while the majority of the items do not receive the same attention. The recommendation algorithms are biased towards these popular items. Popularity bias is an algorithm's tendency to mitigate existing popularity differences between items. It has been proved that popularity bias can lead to certain problems in recommendation, such as alteration of users' preferences to popular items as time goes by or even lead to homogenization of groups of users.

Recommender systems that use collaborative filtering usually focus on popular items rather than on the items of the long tail, which are popular in small user groups. This bias is inherited by all datasets, which include a small number of popular items (known as the long tail). Since the short head items are already popular, recommender systems are presented as smart tools, which can promote non popular items more so that they can increase their popularity. Even though popular items are usually good recommendations, it is also probable that they are widely known. So, generating recommendations only for popular items does not help promote the discovery of new items and helps ignore the interests of niche users. Even though popular items consist of 20 % of the items of a system, more than 80 % of the ratings are provided to them. Since short head items are already well known to many users, this shows if a recommender system is

capable of recommending beyond these items.

Moreover, it may be unfair for providers of less popular or newer items, since these will get less recommended to users. Also, recommending items of the long tail can be regarded as a social good. A purchase under the influence of popularity bias lacks chances of discovering hidden items and gets ruled by a small number of big brands or very well-known artists. A purchase such as this would be more homogenized and would provide fewer chances for novelty and inventiveness. Long tail items are, also, important in order to fully comprehend the users' preferences. Systems that learn actively to discover every user's profile should recommend additional items of the long tail, because a user would not know these items and may even discover deeper users' preferences.

2.2.2 Exposure bias

Machine learning models make a lot of assumptions so that they can recommend recommendations without bias. One of these assumptions is the event in which data is collected randomly. Often, in a machine learning project, humans are those responsible for the collection and labelling of the data. For instance, the problem of constructing a model with machine learning that tries to rank pictures of cats and dogs begins by collecting labelled data of the pictures is a classic problem. The pictures of dogs and cats are randomly gathered and get labelled by an expert or an oracle to generate labelled training and test data.

Recommender systems should be treated differently, because of the way data is collected. It is true that human behaviour is responsible for collecting and labelling data. This happens, because most recommender systems collect interest data or usage data, like views, clicks or ratings by users so that they can generate predictions. The users see the recommendations and provide the next ratings. This feedback loop limits the available data only for items a user was exposed to.

In order to create a recommender system without bias, the data to which a user has been exposed to should be randomly collected and then the user should rate all the items they have seen. However, this is not realistic. This procedure can cause grave problems for users and items. For example, a user might come across a filter bubble problem. If the model cannot learn all of

the different preferences of a user, it may continue to provide the same type of recommendation over and over. What's more, items in a recommender system may suffer from less exposure. The feedback loop can cause an unjust exposure between different items, something that could lead to distorted distribution of ratings and thus create a small number of popular items and a vast number of non-popular or items that have been exposed poorly.

2.2.3 Position bias

Position bias is observed mainly on Information Retrieval. It describes a human tendency to observe or interact with items in certain list positions, despite the actual relevance of the items. Eye tracking research has shown that users are less likely to look at items that rank lower in vertical lists, examining only the first few search results. 65 % of users interact with lists by clicking only the first result that may look relevant, without evaluating the whole list. Position bias creates challenges in the evaluation of recommender systems based on the interaction of users with recommendations. Anyone could understand that the relevance of groups of recommendations can be done by guessing from a users' clicks in a recommendation list. The amount a recommender system is effective can be estimated by using this data. Because of position bias, however, the possibility of a user interacting with an item may not show perfectly the relevance of an item in a recommendation list. Ratings, which are based on data collected by users' clicks, but that do not take the underlying bias into account, may be deceptive.

Moreover, position bias generates results that rank high positions in the results of a search. These are more likely to be clicked by a user, even if they are not that relevant with the user's search. Algorithms that correct position bias typically induce that all relevant results have small possibility of being observed by a user and focus on increasing the relevance of the results that rank low in a recommendation list. However, users rarely have the chance to observe all relevant results, either because the system chose to show a reduced list or because the users do not dedicate any time to observe literally hundreds of thousands of results. In this case, results that rank low, but are relevant, have zero possibility of getting observed by a user and never have the chance to rank high in the recommendation list. This leads to position bias when it comes to results coming

from clicks.

2.2.4 Conformity bias

The basic idea is based on the hypothesis that some of the interactions of users with popular items should not be due to their preferences, but they rather get attributed to a conformity tendency, which amplifies popularity bias. For example, when a user knows about a popular product, even though they might not particularly like it, curiosity and conformity coming from the product’s popularity make the user buy it. The interaction caused by factors that force the user to comply and the interaction caused by personal preferences should not play any role in calculating an algorithm for a recommender system.

2.3 How to create a fair and unbiased recommender system

Strategies that help develop fair recommender systems fall under three categories: pre-, in- and post-processing of a recommender system. Pre-processing transforms training data into being less biased. In-processing procedures turn the training model’s target into one that can be incorporated in a fair recommender system, often adding restrictions or regulation fees, even though many of these methods do not respond well to large datasets. Finally, post-processing approaches ensure that the system will be fair. Post-processing approaches are very attractive in industrial practice, because they do not demand changes in an existing model. Virtually, any model may easily adapt to a post-processing algorithm in order to achieve a fairness objective.

There are five (5) papers on how to create a fair recommender system, which present ways to promote fairness in recommender systems:

2.3.1

Description of the article “A unified optimization toolbox for solving popularity bias, fairness and diversity in recommender systems” written by Sinan Seymen,

Himan Abdollahpouri and Edward C. Malthouse, presented at the MORS workshop, which took place at the 15th ACM Conference on Recommender Systems in Amsterdam in 2021

Results

The writers of the article used the Movielens 1M dataset, because it contains the genre of every movie, something useful for the diversity of the model. The system works on an Intel® Core™ i7-8750H CPU @ 2.20GHz 2.21 GHz processor and a 16GB RAM memory. They used the Gurobi software with optimization gap of 10-4. The Gurobi software uses the B&B algorithm (branch&bound) with time complexity in the worst-case scenario. It, also, uses heuristic methods on that algorithm, increasing the time performance.

The writers utilize the SVD algorithm for estimating the ratings for each item-user pair. The optimization problems can be solved as a post-processing step with the use of the predicted ratings matrix. All the models of the system can be solved using that same matrix. The writers have set the top-k list's size using k (k=10).

Popularity bias

The authors compare the system's second model technique with the xQuAD approach, one of the most effective approaches to control popularity bias. Figure 1 shows the average recommendation popularity and average diversity of the second model and the xQuAD approach for different values of α . From left to right α changes from 0.03 to 0.16 in increments of 0.01. Larger values of α give the same result because constraint 4 is not significantly satisfied. Also, with larger values of α it is observed that the Pop-Opt model performs similarly to the xQuAD model with slightly better score predictions. In terms of average diversity, the authors' model is much better than xQuAD when α takes large values. Therefore, the Pop-Opt model achieves comparable performance to the xQuAD approach in reducing the popularity bias.

Combined models results

The algorithms working for several metrics work well for any metric, but

the Comb-Opt model achieves high performance for metrics ILS, Z and ARP, without reducing the average rating value. For fairness metric for the items, the writers have removed the upper bounds and for the FairRec model, as well as for the Comb-Opt model, except for the solutions with superscript, they beat the results of the FairRec model for all metrics, except for the results of the ARP metric of the FairRec model. Generally, the Comb-Opt model seems to improve all metrics simultaneously. If anyone wants to reduce popularity bias, fairness and diversity, they can combine all of the constraints on that paper. If a user is not interested in one of these metrics, they can remove these constraints and this can result in improving the rest of the metrics (Seymen, Abdollahpouri, and Malthouse 2021).

<i>Model</i>	<i>w</i>	<i>y</i>	<i>α</i>	<i>ILS</i>	<i>Z</i>	<i>ARP</i>	<i>Avg.Rating</i>	<i>Agg. Div.</i>
<i>Comb-Opt</i>	1	0.9	0.09	0.142	0.969	0.067	4.128	1.0
<i>Comb-Opt*</i>	6	0.9	0.055	0.129	0.999	0.053	4.055	1.0
<i>Comb-Opt</i>	7	0.1	0.085	0.080	0.668	0.085	4.672	1.0
<i>Comb-Opt</i>	7	0.5	0.08	0.089	0.843	0.079	4.421	1.0
<i>Comb-Opt</i>	7	0.1	0.08	0.080	0.668	0.080	4.669	1.0
<i>Comb-Opt</i>	7	0.5	0.05	0.092	0.838	0.050	4.393	1.0
<i>Comb-Opt</i>	7	0.99	0.05	0.104	0.994	0.050	4.017	1.0
<i>Comb-Opt</i>	8	0.8	0.1	0.058	0.934	0.076	4.206	1.0
<i>Comb-Opt</i>	9	0.1	0.08	0.022	0.658	0.080	4.653	1.0
<i>Top-k</i>	-	-	-	0.118	0.630	0.155	4.707	0.282
<i>FairRec</i>	-	0.8	-	0.143	0.946	0.077	4.133	1.0
<i>FairRec</i>	-	0.99	-	0.147	0.998	0.054	3.945	1.0
<i>FairRec*</i>	-	0.99	-	0.147	0.999	0.052	3.936	1.0
<i>xQuAD</i>	-	-	-	0.103	0.629	0.089	4.678	0.303
<i>xQuAD</i>	-	-	-	0.118	0.634	0.147	4.707	0.291
<i>WS</i>	-	-	-	0	0.698	0.118	4.536	0.351
<i>WS</i>	-	-	-	0.118	0.634	0.147	4.707	0.291

Table 1: Results of Comb-Opt with different parameter choices

Description of the article “Alleviating the unfairness of recommendation by eliminating the conformity bias”, written by Zhe Ma and Qiang Dong, presented in the Journal of Physics:Conference Series on 2021.

Results

In both datasets, the suggested method is better than the other three methods in accuracy and fairness at the same time. The suggested method exhibited

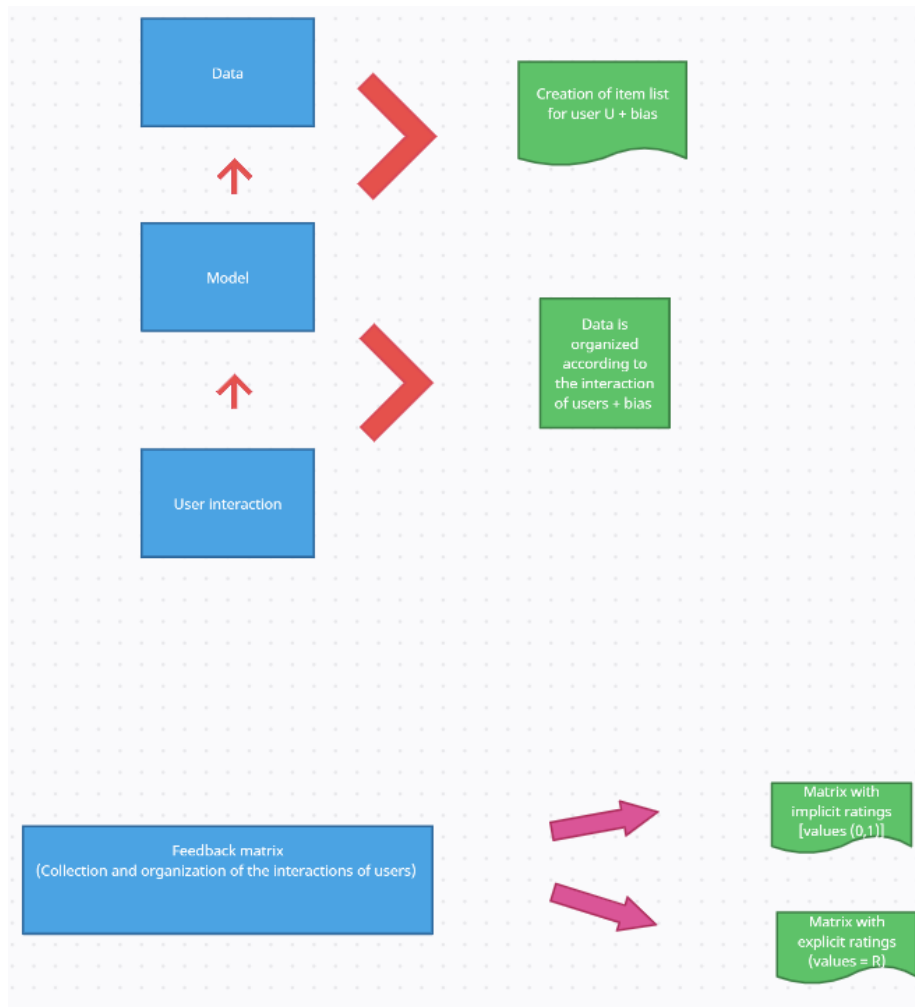


Figure 1: Work model (Alleviating the unfairness of recommendation by eliminating the conformity bias)

34.4 % better results compared to the PD algorithm and 46.14 % compared to the RP3 algorithm for Gini coefficient on the MovieLens dataset, with a smaller increase in accuracy. On the Netflix dataset, the suggested method showed 42.16 % better results compared to the PD algorithm for Gini coefficient. Moreover, compared to Hamming distance, which exhibited a value of 0.8948% for PD algorithm and 0.8731% for RP3 algorithm for the MovieLens dataset and a value of 0.70672% for PD algorithm and 0.70263% for RP3 algorithm for the Netflix dataset, the suggested algorithm showed better values (0.9293% for PD algorithm and 0.80863% for RP3 algorithm) (Ma and Dong 2021).

Description of the article “Deepfair:Deep learning for improving fairness in recommender systems”, written by Bobadilla Jesus et al., published in the International Journal of Interactive Multimedia and Artificial Intelligence on 2020.

Experiment

Data set

The authors used the MovieLens and Netflix datasets. For the MovieLens set, it contains 836,477 links, consisting of 6,038 users and 3,628 movies. Each link has four parts (user, item, rating, timestamp), meaning that the user has rated a movie at the point in time indicated by the timestamp. The rating ranges from 1 – 5, reflecting the user’s preference for the movie. The higher the price, the higher the preference for the movie. In the experiment, links with a score of 3 or more are retained, and links with a score of less than 3 are removed. After filtering the data, it must be ensured that there are no isolated nodes in the filtered data set, because isolated nodes cannot be introduced into the algorithm. The Netflix set consists of 10,000 users, 6,000 movies and 824,802 links. Data filtering was done in the same way as for the MovieLens set.

Results

The experiments conducted were:

- Item Minority Index (IM) and User Minority Index (UM) distributions
- User Minority Index (UM) comparison between each minority and non-minority group

- Increase in the fairness prediction using the heuristic algorithm
- Increase in recommendations for fairness with the use of the heuristic algorithm
- Improvement of the fairness and accuracy error for the recommendations using the suggested architecture for deep learning

The conclusion to which the researchers have reached is that accuracy and fairness can cooperate with each other, meaning that the suggested method can manage the accuracy error when fairness is increased. What’s more, after the neural network is trained using demographic data, it can predict and create recommendations for the users, without knowledge of their demographic information.

The results show an increase of fairness with a cost, though: the accuracy error. The suggested heuristic approach using machine learning returns results of the same nature (Bobadilla et al. 2020).

Description of the article “Investigating potential factors associated with gender discrimination in collaborative recommender systems”, written by Mansoury Masoud et al., presented in the 33rd FLAIRS International Conference on 2020.

Experimental results

When it comes to accuracy, the male users receive more accurate recommendations compared to the recommendations female users receive and regarding miscalibration, except for the SVD++ algorithm, the male users receive more calibrated recommendations in comparison with those the female users receive. The accuracy of the recommendations is not, also, related to the bias in their profiles. The size of the users’ profiles does not play a significant role when it comes to calibration, too.

Regarding accuracy, the connection of the three factors (profile anomaly, profile entropy and profile size) is much more intense. The more intense the profile anomaly, the lower the accuracy becomes. The user groups with high profile entropy receive more accurate recommendations. Finally, the bigger the profile size, the more accurate the recommendations can get. However, the dis-

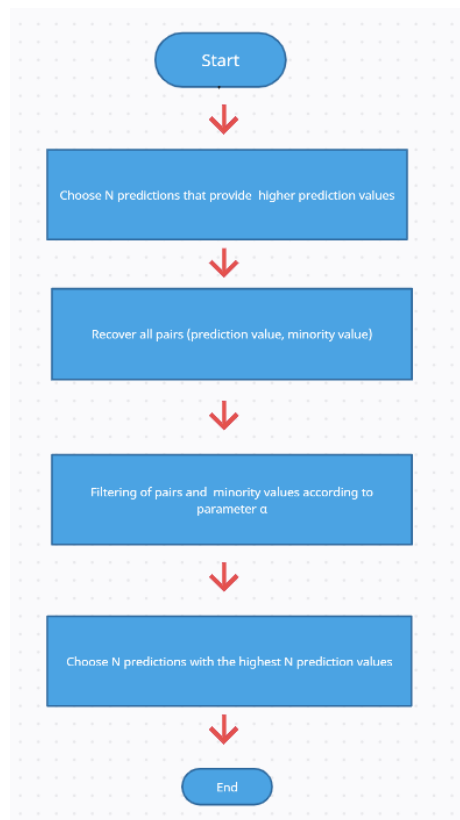


Figure 2: Work model

crimination between male and female users is more evident in algorithms, such as UserKNN, where female users with the same profile size continue to receive recommendations with lower accuracy (Mansoury, Abdollahpouri, Smith, et al. 2020).

Description of the article “Localized fairness in recommender systems”, written by Sonboli Nasim and Robin Burke, published on the Fairness in User Modelling, Adaptation and Personalization Workshop on 2019.

Results

The initial experiments’ results were surprising. With some adjustments, the algorithm designed to be fair concentrated only on a very small fraction of loans, something completely opposite of the expected outcome (diversity of recommendations). Clearly, there was something wrong with the way the protected and not protected groups were identified and, in order to see what was wrong, the researchers began to investigate the data structure with greater detail.

Identification of recommendation needs

The examination of the data demonstrated that the funding rate showed distortions. As it seems, many of the loans were funded quickly. 55 % of the loans would be funded in 10 % of the rate given (within 3 days or less).

The first expectation was that these loans would require small amounts of money, which would need fewer lenders in order to be processed fully. However, there is a connection between the amount of the loan and slow funding. On the other hand, there are loans with a small denominator (loans which do not need many days in order to be fully funded), which do not get funded within the determined deadline and loans with a big denominator (loans that need many days in order to be completely funded), which get funded earlier. So, the PFR function is not the only one that the writers take into consideration for the loan amount.

Since Kiva’s target is to ensure the quick funding of loans, the short head loans do not need any special treatment, because they can find lenders really easily. The researchers’ theory is that these loans come from a few countries with dedicated crowds between Kiva members, which probably have special bonds with these regions. Those dedicated lenders can and want to “grab” lending opportunities to lend money to borrowers in those countries using automatic tools connected with Kiva’s API.

The researchers chose to remove this part of loan distribution (the loans that do not need to be promoted because of their dedicated audience), so that there

will be balance in the availability of fund and to be able to concentrate on the “tail” of the distribution, where the promotion through recommendation could help borrowers fund their loans more quickly.

The writers removed the loans that got funded within 3 days since their introduction in the system by the dataset. They, also, removed the countries with less than 50 loans in a whole year. This subgroup contained 52.645 loans, which were supported by 143.809 lenders for a total amount of \$55.430.175 (Sonboli and Burke 2019).

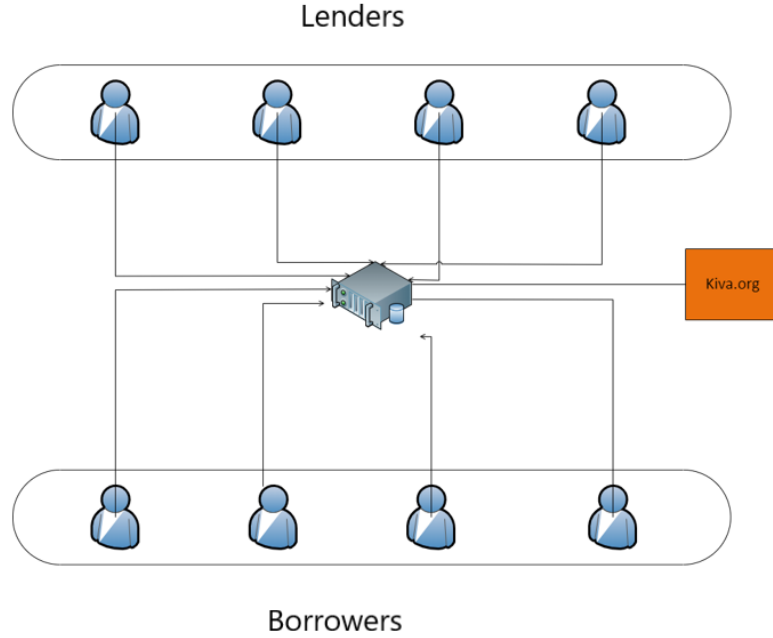


Figure 3: System architecture (Localized fairness in recommender systems)

The figure below shows the general architecture between the lenders, the organization and the borrowers:

Categorization of fair recommender systems

For the purposes of this dissertation, the categorization of the above recommender systems was found to be of critical importance, based on the technologies they incorporate, the algorithms they use and the metrics used to evaluate their performance:

Technologies of fair recommender systems

<i>Technologies/Systems</i>	PD (Ma, 2021)	Deepfair (Bobadilla, 2020)	Mansoury (Mansoury M. A., 2020)	Kiva.org (Nasim Sonboli, 2019)	UOT (Seymen, 2021)
<i>MLN</i>	-	✓	-	-	-
<i>Re-Rank</i>	-	-	-	✓	-
<i>F-aware Opt</i>	-	-	-	✓	-
<i>Constraints</i>	-	-	-	-	✓
<i>PD</i>	✓	-	-	-	-
<i>Bucketization</i>	-	-	✓	-	-
<i>PMF</i>	-	✓	-	-	-

Table 2: Technologies presented in fair recommender systems

On table 2, we can see that the PD system (Ma and Dong 2021), uses PD technology (preferential diffusion), a technique that allows the filtering of data and minimizes popularity bias, so that we can create a fair recommender system. The Deepfair system incorporates MLN and PMF technologies (Bobadilla et al. 2020). MLN (Multilayer neural network) is a technology that uses a multilayer neural network. PMF (Probabilistic matrix factorization) is a technology that is used to infer unknown values for matrices. Moreover, on table 2, it can be seen that Mansoury’s system uses Bucketization: it is a technique through which we can classify items (Mansoury, Abdollahpouri, Smith, et al. 2020). Kiva.org uses Re-ranking and Fairness-aware optimization. Re-ranking technology re-ranks the results of the recommendation and fairness-aware optimization optimizes the results of the recommendation knowing that it should be fair (Sonboli and Burke 2019). Finally, we can see that UOT system uses constraints: a technique that allows the system’s manager to change it, so that they can use their own metrics (Seymen, Abdollahpouri, and Malthouse 2021).

Technologies

Multilayer neural network

Among different types of neural networks, research has focused on Rosenblatt’s perceptrons, known also as multilayer neural networks. In these networks there is a input unit layer, whose only role is to feed input patterns to the rest of the network. Next, there is one or more between or hidden neural layers that evaluate the type of function of the weighted input sum, which send it to the units of the next layer. This procedure goes on till it reaches the final layer,

making the calculation reading possible.

There are no connections leading from a neuron to another, nor are there neurons further from the next level. Once all neurons are updated in the right order, their ranking does not change, meaning that for this architecture time does not play any significant role.

Generally, deep learning models achieve higher accuracy. First, deep learning can use more data. Many traditional machine learning methods present more data. However, when the capacity of the neurons gets increased, the model can improve its performance using more data. Secondly, neural networks are designed to be agile. For example, in order to train a deep learning model with multiple targets, such as “do you want to add this product in the cart?” or “do you want to buy this product?”. Every goal helps the model extract information from the data.

Re-ranking

Some systems tend to model the mutual influences between items to improve the initial list that is given by the ranking algorithm, known as re-ranking. The main idea is to create a rating function by coding patterns in the items. The innovative methods for coding the characteristics’ products are based on RNN, such as GlobalReRank and DLCM. These feed the initial list in a structure that is based on RNN and export the coded product in every step. However, RNN approaches have a limited ability to model the interactions between items in a list. The characteristics’ information of the previous coded item gets degraded along with the coding distance.

Generally, re-ranking as a technique helps the users see items in the list that they wouldn’t be able to see, if they did not appear in the first results of a search. It re-ranks the recommendations so that the user can see these results, which they would not see if they did not look at the whole recommendation list, since most users tend not to look at the whole list, but only the first results the list shows. It exhibits some complexity in comparison to other techniques.

Bucketization

Bucketization is an anonymization technique when someone needs to publish sensitive data. The idea is that the data gets grouped in buckets so that they

can hide the relation between sensitive information and recognition information. Bucketization does not need ranking of the values of the characteristics, thus making it applicable to more datasets. A drawback of previous bucketization techniques is the homomorphic privacy setting and the homomorphic bucket size, something that leads to non achievable privacy for users or leakage of information, if the values are sensitive to variables. This technique ensures the privacy of users' data, because their items are divided into buckets, so these items do not exist in the same server, thus making it difficult for anyone to gain access to them. Even if anyone gains access, they will have access only to a small portion of the data.

Preferential diffusion

It is a technique, in which a user chooses a recommendation from a recommendation list based on their friends' preferences. That means, the user gets influenced by their friends' preferences (diffusion) and they choose a recommendation from the recommendation list based on what their friends like (preference).

Constraints

The constraint framework improves the non-accurate measurements of the recommendations, combining different constraints designed for different measurements. Uniting all of the different problems that are not related to accuracy in recommender systems can benefit the research community and researchers study how they can create a list with k items for each user, assuming that every user's preferences for each item has already been estimated using some existing recommender system.

For all of these problems, this technique maximizes the same objective function: the average ratings in every user-item pair in the recommendations (that is the relevance of the recommendations). Different problems can be solved with the addition of different constraints. So, every problem will have a similar structure, making them usable and comprehensible.

The constraints technique has one advantage: in every problem it is possible to use some constraint, which will solve the problem. For any problem in the

system, any constraint can be used in order to solve it. The disadvantage of this technique is that each time one or more problems may be solved, but other problems surface. For instance, we can use constraints so that recommendations will be accurate, but at the same time the system will not be fair and vice versa.

Probabilistic matrix factorization

There are many matrix factorization models for collaborative filtering that learn from data that is not missing at random (MNAR – Missing at not random). Matrix factorization models present an excellent prediction ability. However, these models often assume that data is missing at random (MAR – Missing at random) and this happens quite rarely. For example, data is not missing at random if users rate the items they like more than those they do not like. When the hypothesis that data is missing at random is false, results can be biased and the predictive performance can suffer.

Usually, most users rate only a small fraction of the available data, so most data for collaborative filtering are missing. Matrix factorization models have become popular in collaborative filtering, because 1) they can be powerful, they can overadapt and they can automatically make estimations, 2) they can adapt to different kinds of data, such as continuous, binary or tactical data and 3) they can theoretically manage the missing data. A common hypothesis that these models do is that collaborative filtering data is missing at random.

Fairness-aware optimization

This technology works as such: it uses datasets with balanced user neighborhoods. In other words, this technique assumes that there should be users who represent different user groups, in order to work and generate fair recommendations. For example, it needs an equal number of male and female users in a system. This, however, is not possible, because there is a big diversity of users in recommender systems.

Regarding the above described technologies, the best technology is probabilistic matrix factorization. This technology can make really good predictions

over the users' preferences. It can handle the missing data, that is the values of the matrices that are missing, therefore have not been rated. It can predict these values with very good accuracy and succeeds in generating accurate and fair recommendations. The rest of the technologies described do not achieve this goal, either partly or not at all. Based on the above, the system that is better and fairer than the rest is Deepfair.

Algorithms of fair recommender systems

<i>Algorithms/Systems</i>	PD (Ma, 2021)	Deepfair (Bobadilla, 2020)	Mansoury (Mansoury M. A., 2020)	Kiva.org (Nasim Sonboli, 2019)	UOT (Seymen, 2021)
<i>PD/TS-PD</i>	✓	-	-	-	-
<i>UserKNN</i>	-	-	✓	-	-
<i>ItemKNN</i>	-	-	✓	-	-
<i>SVD++</i>	-	-	✓	-	✓
<i>ListRankMF</i>	-	-	✓	-	-
<i>F-aware algorithm</i>	-	-	-	✓	-
<i>GDA</i>	-	✓	-	-	-
<i>BackPropagation</i>	-	✓	-	-	-
<i>Heuristic</i>	-	✓	-	-	-

Table 3: Algorithms used in fair recommender systems

On table 3, it seems that the PD system uses the PD algorithm, which is an abbreviation for preferential diffusion. The Deepfair system utilizes the GDA, Backpropagation and heuristic algorithms. The Gradient descent algorithm is an optimization algorithm used to train machine learning models and neural networks. Here's how it works: it calculates the next point to which it will go using the gradient of that point. After subtracting the value of the gradient, it jumps to the next point. The backpropagation algorithm works in a similar way as the previous algorithm. Finally, heuristic algorithms are algorithms that sacrifice elements, such as accuracy, for speed. They can solve a problem quickly, but not optimally.

Mansoury's system uses UserKNN, ItemKNN, SVD++ and ListRankMF. User and ItemKNN are variants of the nearest neighbour algorithm. Here's how they work: they use proximity to make assumptions about ranking the items. They are commonly used for ranking items and the idea behind the way they work is that similar items/users can be found next to each other. The difference between these variants is that the UserKNN algorithm suggests the k-nearest

similar users to a user, while ItemKNN suggests the k-nearest similar items to a user. The SVD++ algorithm (Singular Value Decomposition) is an algorithm, which attempts to guess empty matrix cells' values, increasing the accuracy of the recommendations. ListRankMF is an approach used for ranking the results of a search. Moreover, on table 3 it can be seen that Kiva uses a fairness-aware algorithm. This is an algorithm that is based on the assumption that recommendation and the whole system should behave in a fair manner. UOT system uses the SVD++ algorithm.

Algorithms

User and Item KNN

The k Nearest Neighbors (kNN for short) algorithm is a “lazy” learning method. It has been considered to be one of the simplest machine learning algorithms. The logic of kNN is that similar samples belonging to the same class have a high probability of being similar, while the basic idea of the algorithm is to first select the k nearest neighbors for each test sample. A disadvantage of the algorithm is that it is sensitive to the choice of the value of k. Although studies have focused on this topic for a long time, the selection of the k value in the kNN algorithm is still very difficult and controversial.

The element-based kNN algorithm is a neighbor-based approach that belongs to the class of memory-based cooperative filtering algorithms. Instead of calculating the similarity of users, the algorithm calculates the similarity of items that the target user has rated (purchased or noticed) with the target item.

SVD and SVD++

Singular Value Decomposition (SVD) is a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. SVD is a matrix factorization tech-

nique, which reduces the number of characteristics of a data set by reducing the dimensionality of the space from N -dimensional to K -dimensional (where $K < N$). In the context of recommender systems, SVD is used as a collaborative filtering technique. It uses a table structure where each row represents a user and each column represents an item. The items in this table are the ratings given to items by users.

The factorization of this matrix is done by using singular value decomposition. It finds matrix factors from the factorization of a high-level matrix (user-item-score). Singular value decomposition decomposes one matrix into three other matrices.

This method helps make the recommendation system fair, because it reduces the dimensions of the original matrices to 3 smaller, so the values that the system did not know before, i.e. the values that were missing from the matrices, are no longer a problem and the algorithm can generate fair and accurate recommendations for users, since it no longer needs to know these values.

ListRankMF

ListRankMF is a list-ranking algorithm for recommender systems. It takes as input an matrix of known ratings given by users to certain items and produces as output two matrices, one for users and one for items. Each of these tables has a (latent) feature vector for each user and each item. A user's predicted rating of an item is then given by the inner product of their feature vectors. These scores are only used to rank the recommendations. They only serve the purpose of sorting items by preference for each user. Using this, one can generate recommendations.

This technique is similar to the re-ranking technology, since both technologies are recommendation ranking technologies. The first,

however, presents low complexity.

Gradient descent algorithm

The gradient descent (GD) algorithm is one of the most common optimization algorithms and the most popular neural network optimization method. It works to find a distinct role in a local minimum. It is used to find the values of the parameters of a function that reduce the loss function as much as possible. To find a local minimum of a differential equation iterative steps must be taken in the opposite direction of the gradient in the current stage of the function because this is considered the steepest descent direction. On the other hand, stepping in the direction of the slope will result in a local maximum of this characteristic. Then the approach is referred to as a step climb.

This technique aims to optimize the objective function in the recommender system, which is one of the main goals in recommender systems. In other words, the goal is to optimize the function so that it somehow finds the ratings for the items that the user has not rated. Thus, the system becomes fairer, since there are now values for each item of the system. So, the system can estimate the user's preferences and suggest better and fairer recommendations.

BackPropagation algorithm

The Backpropagation algorithm searches for the minimum of the error function in the weight space using the gradient descent method. The combination of weights that minimizes the error function is considered as a solution to the learning problem. Since this method requires the computation of the gradient of the error function at each step, it must guarantee the continuity and differentiability of the error function.

The idea behind the Backpropagation algorithm is quite simple: the output of the neural network is evaluated against the desired output. If the results are not satisfactory, the connection (weights) between the layers is modified and the process is repeated again and again until the error is small enough.

This technique is also quite good at making the recommendation system fairer, but it already uses the gradient descent algorithm. Therefore, this entails higher computational costs for the system. That is, the system becomes fairer, but more computing power is required from the system.

Heuristic algorithms

In relation to the quality issue, the goal of the heuristic algorithm is to find the best possible solution for all sides of the problem. Algorithms that either give almost the correct answer or provide a solution that is not sufficient for all aspects of the problem are called heuristic algorithms. This group includes a wide range of methods based on traditional techniques. The simplest of the search algorithms is the exhaustive search which tries all possible solutions from a predefined set and then selects the best one.

A heuristic is a technique designed to solve a problem faster when classical methods are too slow, or when we want an approximate solution, when classical methods fail to find an exact solution, but they don't. They do not guarantee that the best solutions will be found, so they can be considered as approximate rather than exact algorithms. These algorithms usually find a solution close to the best and find it quickly and easily. Sometimes these algorithms can be exact, that is, they actually find the best solution, but the algorithm is still called a heuristic until this best solution is proven to be the best.

Fairness-aware algorithm

This algorithm is an extension of the Sparse Linear Method (SLIM). It is a generalized algorithm of the item-based recommendation class in which a regression coefficient is learned for each user u , item i pair. The algorithm uses regularization to control how different neighbors are weighted, aiming to achieve a balance between protected and unprotected neighbors for each user.

Regarding the algorithms presented above, the most efficient algorithm is ListRankMF. It is the most efficient, because it is very simple and does not consume much computing power or time, but mainly because it reorders the recommendations, so that the user sees recommendations that they would not have seen because of their position in the list. So, it achieves the purpose of fair recommender systems, i.e., to make recommendations fair, since by re-ranking users will also take into account recommendations they didn't know existed. Gradient descent and SVD algorithms are not better, because they randomly fill in the missing values from the user-item score tables, so it is not guaranteed that the values they fill in will be correct. So, just because the tables have random values, doesn't mean they will generate fair recommendations.

The UserKNN and ItemKNN algorithms assume that users or items will be similar to users or items that are already known. Heuristic algorithms have many advantages, but, also, disadvantages. One of them is their time complexity and consumption of computing power to find a solution. This makes them unsuitable. Finally, the algorithm that is fairness aware, that is, it takes into account the fact that the system should be fair, but what this algorithm does not take into account is the fact that there can be

no balance in terms of users. It is not possible to have a balance between the gender of the users, for example, it is not possible in a system to have an equal number of women and men. So, since it is not possible to achieve something like this, due to the diversity of users, the algorithm cannot in fact be fair. Regarding the algorithm used by each system, the one using the fairest algorithm is the system described in the work of Mansoury et al.

Metrics of fair recommender systems

<i>Metrics/Systems</i>	PD (Ma, 2021)	Deepfair (Bobadilla, 2020)	Mansoury (Mansoury M. A., 2020)	Kiva.org (Nasim Sonboli, 2019)	UOT (Seymen, 2021)
<i>Precision</i>	✓	-	✓	-	-
<i>Coverage rate</i>	✓	-	-	-	-
<i>Gini coefficient</i>	✓	-	-	-	-
<i>Hamming distance</i>	✓	-	-	-	-
<i>Miscalibration</i>	-	-	✓	-	-
<i>PFR</i>	-	-	-	✓	-
<i>ILS</i>	-	-	-	-	✓
<i>Z</i>	-	-	-	-	✓
<i>ARP</i>	-	-	-	-	✓
<i>Agg.Div.</i>	-	-	-	-	✓
<i>Relu activation</i>	-	✓	-	-	-

Table 4: Metrics used in fair recommender systems

Here is an explanation of the metrics that measure the performance of fair recommender systems: The precision metric refers to the ability of a model to classify only relevant data. Analogously, it refers to the ratio of true positive to true positive + false positive values. The Coverage rate metric is a metric for measuring the performance of recommendations in terms of how fair the system is. The Gini coefficient metric is a metric that works better than the previous one (coverage rate). The Hamming distance metric is a metric that measures how personalized a list of recommendations is, i.e., if it is different, if it does not contain similar recommendation results.

The miscalibration metric: this is a metric that shows that the recommendations are not fair to the users. The higher the value of this metric, the more unfair the recommendations. The PFR metric (Percentage funding rate): shows the percentage with which a system satisfies the needs of its users. The Intra-list similarity (ILS) metric is a metric that measures how similar two items in a recommendation list are and is used to maintain diversity in the recommendation system. The Z metric: this is a metric related to how fair the recommendation system is, but, also, whether the producers' products are recommended equally fairly for all producers. Example for the Z metric: The authors define the Z metric as the disparity in exposure between providers. The metric takes the value 1 when providers' exposure to the system is fair, that is, when their products are exposed equally and fairly to all users. The metric formula is the following:

$$-\sum_{s \in S} \binom{Rs}{|U|k} \log |S| \binom{Rs}{|U|k}$$

The formula is broken down as follows:

$$-\sum_{s=1}^3 \binom{Rs}{|U|k} \log |S| \binom{Rs}{|U|k} \leftrightarrow -\left[\left(\frac{R1}{|U|k}\right) \log 1 \binom{R1}{|U|k} + \left(\frac{R2}{|U|k}\right) \log 2 \binom{R2}{|U|k} + \left(\frac{R3}{|U|k}\right) \log 3 \binom{R3}{|U|k}\right]$$

Here is an explanatory table of the elements of the formula:

Explanatory table of elements of the formula Z
s = number of items
S = number of groups of items
Rs = total number of recommendations from group s
 U k = items of the system recommended to the user

Table 5: Elements of Z metric

Based on the above data, the formula for the metric Z is analyzed. The values of the fields are random. Here is a table of values for the example:

Table of values for metric Z	
Rs	10/13/15
 U k	5 3, 20 1, 7 4
Log (S)	20 (fixed value)
Σ	1-3

Table 6: Values for Z metric

$$\begin{aligned}
& - \sum_{s=1}^3 \left(\frac{10}{|5|3} \right) \log 20 \left(\frac{10}{|5|3} \right) \leftrightarrow - \left[\left(\frac{10}{|5|3} \log 20 \frac{10}{|5|3} \right) + \left(\frac{13}{|20|1} \log 20 \frac{13}{|20|1} \right) + \left(\frac{15}{|7|4} \log 20 \frac{15}{|7|4} \right) \right] \\
& \leftrightarrow [(0,6 \cdot 1,3 \cdot 0,6) + (0,65 \cdot 1,3 \cdot 0,65) + (0,53 \cdot 1,3 \cdot 0,53)] \leftrightarrow [0,468 + 0,54925 + 0,36517] \\
& \leftrightarrow 1,38242
\end{aligned}$$

According to the results, the exposure of the providers in the system is not fair, since the metric took a value above 1.

The ARP (Average recommended popularity) metric is related to the innovation of a fair recommendation system. That is, the smaller the values, the more innovative a recommendation system is.

The ARP (Average Recommended Popularity) metric measures the average popularity of an item. The lower the value of this metric, the better, since low values indicate the novelty of the system and lower popularity of popular items. Its formula is as follows:

Table of values for ARP metric	
j	1-3
i	1-3
ω_i	$\Omega_1=0,02, \Omega_2=0,05, \Omega_3=0,0125$
L_j	$L_1=10, L_2=7, L_3=12$
U	3 (fixed value)

Table 7: Values for ARP metric

$$ARP = \frac{1}{|U|} \sum_{j \in U} \sum_{i \in L_j} \frac{\omega_i}{|L_j|}$$

L_j is the set of all items user j has been recommended. ω_i measures the popularity of an item i , as the ratio of the number of ratings received by this item to the aggregated number of ratings of every item that appears in the system. The values in the applied example are random. Here is a table of values:

The above formula is analyzed as follows:

$$\begin{aligned} ARP &= \frac{1}{|U|} \sum_{j \in U} \sum_{i \in L_j} \frac{\omega_i}{|L_j|} \leftrightarrow \frac{1}{|U|} \sum_{j=1}^3 \sum_{i=1}^3 \frac{\omega_i}{|L_j|} \leftrightarrow \frac{1}{|U|} \left[\left(\frac{\omega_1}{|L_1|} + \frac{\omega_2}{|L_2|} + \frac{\omega_3}{|L_3|} \right) + \left(\frac{\omega_1}{|L_1|} + \frac{\omega_2}{|L_2|} + \frac{\omega_3}{|L_3|} \right) + \left(\frac{\omega_1}{|L_1|} + \frac{\omega_2}{|L_2|} + \frac{\omega_3}{|L_3|} \right) \right] \\ &\leftrightarrow \frac{1}{|3|} \left[\left(\frac{0,02}{|10|} + \frac{0,05}{|7|} + \frac{0,0125}{|12|} \right) + \left(\frac{0,02}{|10|} + \frac{0,05}{|7|} + \frac{0,0125}{|12|} \right) + \left(\frac{0,02}{|10|} + \frac{0,05}{|7|} + \frac{0,0125}{|12|} \right) \right] \\ &\leftrightarrow \frac{1}{3} [(0,002 + 0,007 + 0,001) + (0,002 + 0,007 + 0,001) + (0,002 + 0,007 + 0,001)] \leftrightarrow \\ &0,33 + [(0,01 + 0,01 + 0,01)] \leftrightarrow 0,33 + 0,03 = 0,36 \end{aligned}$$

The resulting value is very small, which shows that the metric works correctly for its purpose, since with this metric it is possible to measure whether the system is fair and novel, because through novelty the reduction in popularity of the most popular system components.

The metric Agg. Div. (Aggregate Diversity) represents the number of unique recommendations. Finally, the metric relu activation (rectified linear unit): is a linear function, which produces results when they take a positive value, otherwise it produces no result.

Description of the article “A study of position bias in recommender systems”, written by Collins Andrew et al, published in the book Transforming Digital Worlds on 2018.

Results

In this article, the writers present CTR for non-mixed groups of data, as expected of a complete recommender system. The items with higher rank have a higher CTR value (0.243 % on average for Sowipor, 1.281 % on average for Jabref) and a lower CTR value as the rank decreases. The users should expect relevant results of a certain degree from an effective recommender system, so it is not weird to see CTR decrease as ranking decreases.

If a large number of recommendation groups mix smoothly and if the users examine the lists without bias, it can be expected that CTR for each rank will be equal. When users are not prejudiced against recommendations, CTR gets a value of 0.123 % for Sowipor and 0.683 % for Jabref. CTR can be calculated as the ratio of total clicks on recommendations to the total number of recommendations given. Some smaller researches about position bias on recommender systems have found that after recommendations are mixed smoothly, CTR is almost equal for each ranking.

However, on this paper, the data found from mixing recommendations show an important relationship between recommendation ranking and if a user decides to click an item or a recommendation (the chi-squared test showed that $p < 0.01$). CTR for mixed groups of recommendations can be reduced with the same rate as the rate for non-mixed recommendation groups).

There are some similar results for users on Jabref, with a small difference between the mixed and non-mixed groups of recommendations. In both cases, the position bias is obvious as shown by the

users' behaviour (the way they click on recommendations). It seems that the users do not interact with recommendations in a logic way.

When mixed data is analyzed according to the position the most accurate recommendation has ranked, the users' behaviour is clearer. Every analysis seems to contain position bias, meaning that the items in higher ranks still have a higher CTR and items on lower ranks have a lower CTR, even though the recommendations get mixed up randomly. However, on average, despite ranking, people search and can discern the relevant items. CTR gets higher values for all lists containing the most relevant item, with a 29 % increase on average (Collins et al. 2018).

Description of the article “All the cool kids, how do they fit in? Popularity and demographic biases”, written by Ekstrand D. Michael et al., presented on the Conference on Fairness, Accountability and Transparency in 2018.

Experimental Protocol The writers divided each dataset with 5-fold cross validation. The main results were extracted with the chosen sampling strategy of LensKit based on users: the writers choose five test sets of users’ data and five ratings for testing for each user. The rest of these users’ ratings along with the users’ ratings except the test set construct the training set for that test set. For the LFM360K dataset, the researchers took samples of five discrete sets of 5000 users for tests, so that every test set will reduce the calculation time. For LFM1K and MLM1 datasets, the writers divided the users in five separate groups.

They, also, tested Bellogin’s U1R method to eliminate popularity bias. This method chooses the test sets of the items instead of choosing the test sets of the users and creates a different recommendation list for each user-item pair. The idea behind this method is that by having the same number of ratings for each item, recommender systems that favor the popular items, cannot “win” just by demonstrating only popular items instead of not popular ones.

Results The writers conducted an experiment that considers the performance of traditional recommendation algorithms for different age and sex groups accordingly, in order to quantify the extent to which demographic information affect the total user satisfaction.

For each algorithm that exhibits a better performance on each test set, the writers compared the differences in utility for each de-

mographic group. The ML1M and LFM360K datasets demonstrate significant differences between sex groups and the LFM360K dataset shows differences between age groups.

Reduction of popularity bias In order to reduce popularity bias, the researchers ran the ML1M version of the dataset, using U1R Bellogin’s protocol. Since this protocol divides the items and not the users, different users could have different values of test items and the distribution of demographic items of users could be different than the underlying data.

The writers compared two average strategies: creating an average between every user-item pairs for every sex and creating an average for the results of the recommendations for all users and noticed no actual difference.

Age “tells” a different story – for the LFM360K dataset, we can observe a different pattern when distributing accuracy between the different ages compared to what we would observe under the evaluation protocol. It is not clear which pattern provides a clearer image, but this indeed shows that attempting to correct a problem (popularity bias) could affect the results for another problem (demographic bias) (Ekstrand et al. 2018).

Description of the article “Feedback loop and bias amplification in recommender systems”, written by Mansoury Masoud et al., presented at the International Conference of Information and Knowledge Management in 2020.

Results Increase of popularity bias: The recommendation models can make popularity bias more intense on input data over the course of time because of the feedback loop phenomenon.

Despite the fact that these algorithms begin with different values of average popularity, they all exhibit an incremental tendency when it comes to average popularity over many iterations. The slope for BPR algorithm seems to have a more rugged when compared with other algorithms, showing that this algorithm increases bias. The true reason for these discrepancies on performance on different algorithms needs to be further researched. As the recommender system focuses more on items that are popular, it will recommend less long tail items. The increase of bias through many iterations could create two more problems:

- Changing the picture of what a user prefers
- And the rule of one user group (majority group) over another group (minority group), which might eliminate discrepancies between groups and bring homogenization.

In all algorithms that generate recommendations, we can notice that the divergence of profiles of users from their original preferences is incremental over the course of time. It is worth noting that the change in users' preferences is the change of those preferences by the system and not a real change of their preferences. This change on the representation of the users' preferences in the system has consequences. One of those is that recommender models may be unable to apprehend the real preferences of the users when they create recommendations for them (Mansoury, Abdollahpouri, Pechenizkiy, et al. 2020).

Description of the article “The unfairness of popularity bias in recommendation”, written by Abdollahpouri Himan et al., presented in the RMSE Workshop, which was conducted along with the ACM 13th International Conference on Recommender Systems in 2019.

Algorithmic propagation of popularity bias

There is bias in the data of the ratings: specific items are frequently rated, while other items get rated only by a few users. The writers observe the overall performance of the system, without focusing on how it works for different users. They put many algorithms to the test, that include UserKNN, ItemKNN, SVD++ and BiasedMF. For all of these algorithms, they used 80 % of the data for training and 20 % for testing. For each user in a test set, a list of 10 items is generated. The researchers have, also, adjusted the algorithms so as to achieve similar accuracy (around 0.1) in order to get a fair comparison of the increase of popularity bias. They, also, have used two simple algorithms, MostPopular and Random.

It seems that in every algorithm, except for the Random algorithm, many items never get recommended.

MostPopular algorithm has the strongest connection between how many times an item gets recommended and how many times it gets rated, followed by UserKNN and ItemKNN, as well, which also have a strong connection. Looking at the BiasedMF algorithm, there is not any connection between how many times gets rated and how many times it gets recommended. The SVD++ algorithm exhibits a positive relation between how many times an item gets rated and how many times it gets recommended. The Random algorithm shows no correlation. For each point in the algorithm section, one can see the average score for the corresponding element to be

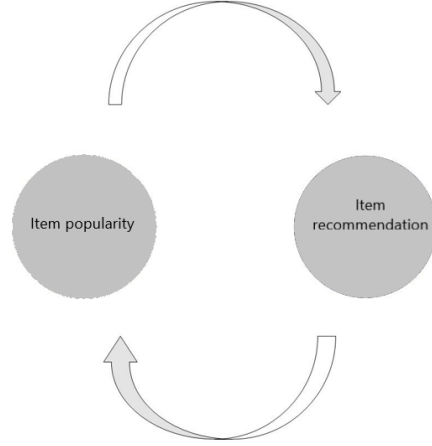


Figure 4: Algorithmic propagation of bias (The unfairness of popularity bias in recommendation)

able to evaluate the quality of the proposed elements using each algorithm. Here is a figure with the algorithmic propagation of biases:

The authors then define Group Average Popularity (GAP (g)), a metric that measures the average popularity of items on user profiles in a given group g or in their recommended lists. If one looks at the recommendations suggested to a group of users, then GAP measures the mean popularity of the user items in that group. On the other hand, if one looks at user profiles, then GAP gauges the mean popularity of the rated items by the users in that group. Here is the GAP calculation formula: GAP:

$$left(right) = \frac{\sum_{u \in g} \frac{\sum_{i \in P_u} \phi(i)}{|P_u|}}{|g|}$$

Where g is the group of users (N, D or B).

$$\varphi(\cdot)$$

is the popularity of a particular item (how many times it is rated against the total number of users) and P_u is the list of items in user u 's profile. The GAP value is represented as $GAP_{-}(P)$ and the GAP for recommendations as $GAP_{-}(r)$. For every recommendation generation algorithm, the authors measure the discrepancy in GAP, which is the amount of unwanted recommendation popularity imposed by the algorithms on every group. A value of 0 of $\Delta GAP=0$ shows that user preferences are fairly represented against popularity of items in recommendations. Here is the formula of the GAP change:

$$\Delta GAP(g) = \frac{GAP(g)r - GAP(g)p}{GAP(g)p}$$

The MostPopular algorithm dictates the highest positive discrepancy in GAP, followed by ItemKNN. The Random algorithm has a negative ΔGAP , meaning that recommendations are on average not as popular as the average popularity of user profiles in every group. Also, it can be observed that Special users (niche) seem to have the largest ΔGAP in every algorithm. These results depict the unfairness of popularity bias in recommendations and how it affects different users depending on their interest in popular items (Abdollahpouri et al. 2019). Here's a diagram showing how popularity bias is compounded:

2.4 Categorizing recommender systems against popularity bias

In this dissertation it was considered important to incorporate a categorization of recommendation systems, which work against popularity bias. This categorization was made based on the technologies they incorporate, the algorithms they use and the metrics that mea-

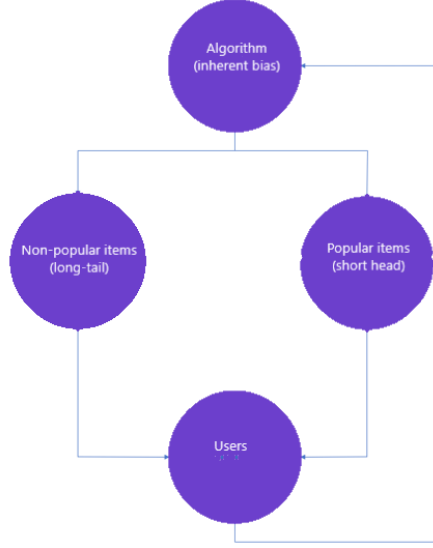


Figure 5: Popularity bias

sure their performance:

Recommender system technologies against popularity bias

<i>Technologies/Systems</i>	Jabref/Sowiport (Collins, 2018)	Ekstrand (Ekstrand, 2018)	Feedback Loop (Mansoury M. A., 2020)	Mansoury (Abdollahpouri, 2019)
<i>RaaS/Mr. DLib</i>	✓	-	-	-
<i>LensKit</i>	-	✓	-	-
<i>5-fold cross validation</i>	-	✓	-	-
<i>FL Simulation</i>	-	-	✓	-

Table 8: Recommender system technologies against popularity bias

In Table 8, it can be seen that the Jabref/Sowiport system uses RaaS (Recommendation as a Service) technology, which recommends items to users (Collins et al. 2018). The Ekstrand system uses the LensKit framework, which is a framework for building and evaluating various algorithms based on collaborative filtering on multiple public datasets with different types of feedback (Ekstrand et al.

2018).

Also, 5-fold cross validation technology is applied: this is a statistical method, which is used to estimate the ability of machine learning models with limited data. According to this technique, the procedure has a parameter called k , which refers to the number of user groups into which a dataset can be divided. When it takes the value 5, then it means that the data set can be divided into five (5) user groups. Table 5 also shows the feedback loop simulation technology in the Feedback Loop system (Mansoury, Abdollahpouri, Pechenizkiy, et al. 2020), which simulates the recommendation generation process by repeating the generation of recommendation lists and updating the user profiles.

Technologies

RaaS / MrDLib

Application providers are tasked with collecting interaction information (feedback) from their users, building models based on that feedback and generating personalized recommendations from their catalog. Mr.DLib's service allows digital libraries and reference managers to easily integrate a recommendation system. The user is not required to make any special effort or have prior knowledge about recommender systems.

How does Mr.DLib work? A contributor requests related articles from MrDLib. Mr.DLib returns a list of related articles to the partner. The partner displays the relevant articles on their website. Before this process can begin, Mr.DLib needs access to the meta-data of the partner's documents.

LensKit

It is a flexible platform for research in recommender systems. LensKit provides carefully tuned implementations of leading collaborative filtering algorithms, APIs for common recommender system use cases and an evaluation framework for performing algorithm evaluations offline.

5-fold cross validation

The numerical value of cross-validation k (k -fold) in predictive machine learning models is an essential element that affects the performance of the model. A correct choice of k leads to greater accuracy, while a poorly chosen value for k can affect the performance of the model. In the literature, the most commonly used values of k are the values of five (5) or ten (10), as they suffer neither from extremely high bias nor from very high variance.

One technique used to assess training models is cross-validation. This is typically used where the machine learning task is prediction and one seeks to assess how accurate a predictive model will perform in training. The goal of cross-validation is to assess the model's ability to classify new data not used to estimate the model to highlight problems, such as bias or overfitting. Additionally, this is done to provide insight into how the trained model will generalize to an unknown data set or real-world problem.

Feedback loop Algorithms used to generate recommendations suffer from popularity bias. Some popular items (short head) are often recommended, while most of them (long tail) get ignored. The users then use these recommendations and their interaction will be recorded and added back to the recommendation system. This is called a vicious feedback loop.

Regarding the technologies used in the tasks, the technology that enables impartiality in recommendation systems is the feed-

back loop. This technology is more effective than the others, because it allows the system administrator to see and predict what will go wrong and where bias will appear in the system, so they can quickly fix it. RaaS (recommendation as a service) is a service, which simply suggests recommendations to users and LensKit is a platform, which is used to implement research on recommender systems. None of these technologies provide evidence of how bias in recommender systems could be reduced. Finally, 5-fold cross validation technology provides feedback on the performance and accuracy of a recommender system. So, the system that minimizes bias in recommender systems is the system described in the paper.

Recommender system algorithms against popularity bias

<i>Algorithms/Systems</i>	Jabref/Sowiport (Collins, 2018)	Ekstrand (Ekstrand, 2018)	Feedback Loop (Mansoury M. A., 2020)	Mansoury (Abdollahpouri, 2019)
<i>Depth-first</i>	✓	-	-	-
<i>Pop</i>	-	✓	-	-
<i>Mean</i>	-	✓	-	-
<i>Item-Item</i>	-	✓	-	-
<i>User-User</i>	-	✓	-	-
<i>FunkSVD</i>	-	✓	-	-
<i>BPR</i>	-	-	✓	-
<i>UserKNN</i>	-	-	✓	✓
<i>ItemKNN</i>	-	-	-	✓
<i>SVD++</i>	-	-	-	✓
<i>Random</i>	-	-	-	✓
<i>BMF</i>	-	-	-	✓

Table 9: Algorithms used in recommender systems against popularity bias

In table 9, it is observed that the Jabref/Sowiport system uses depth-first search algorithms, which searches tree structures with nodes. It sets a random node as the seed and then searches each branch of the tree using backtracking and the Insufficiently Discrete Ranking (IDR) algorithm: this is a technique where the system assumes that the user can see that almost all items are relevant among themselves and to choose the items that are on top of the

list of recommendations).

Also, one can see that in table 8, Ekstrand's system implements Pop, Mean, Item-Item, User-User and FunkSVD algorithms. Pop suggests the items that are most preferred over others. The Mean algorithm suggests the items with the highest average score of the ratings. The Item-Item algorithm is an algorithm that belongs to the category of collaborative filtering. It uses twenty (20) neighbors and cosine similarity. Its version that uses direct feedback from users (i.e., yes or no or when scoring from 1-5) normalizes the scores by subtracting their averages. Its version that uses indirect feedback (i.e., receives information from user clicks or how many times an item has been visited) replaces averages with a sum of similarities. The User-User algorithm is a similar algorithm to the previous one.

The FunkSVD algorithm is essentially the GDA algorithm (Gradient descent algorithm) is an optimization algorithm, used to train machine learning models and neural networks. It works like this: it calculates the next point to go to, using the slope of that point. After subtracting that value it found (of the slope), it jumps to the next point. Also, the table shows that BPR and User KNN algorithms are used in the Feedback Loop system. The BPR (Bayesian Personalized Ranking) algorithm is an algorithm that tries to rank the data for a user, that is, it creates a list of recommendations with Bayesian analysis, a statistical method that tries to make assumptions about a user's preferences based on data in some sample data. UserKNN/ItemKNN algorithms: These are variants of the nearest neighbor algorithm. It works like this: it uses proximity to make assumptions about the classification of the data/items. It is commonly used as an algorithm for sorting items and works on the assumption that similar items/data can be found close to each other. The difference between these two variants is that the first

one suggests the k-closest similar users to the user and the second one suggests the k-closest similar items that the user has set to like.

Finally, Mansoury's system appears to implement User KNN, Item KNN, SVD++, Random and BMF algorithms (Abdollahpouri et al. 2019). The SVD++ (Singular Value Decomposition) algorithm is an algorithm that tries to guess values of empty cells in item matrices, thus enhancing the accuracy of recommendations. The Random algorithm suggests random items of the system. Finally, the BMF (Biased Matrix Factorization) algorithm: this algorithm is essentially a combination of FunkSVD along with the following assumption: many users tend to rate higher than other users, so some of the items will be rated higher than other items. (The User KNN algorithm is described above).

Algorithms

Depth-first search

The search starts by expanding the original node, i.e., creating its successors. At each subsequent step, one of the previously created nodes is expanded until a target node is found. The solution path can be constructed by following pointers backwards from the target node to the origin node. In depth search, one of the most recently created nodes is expanded first. Since the deepest nodes are expanded first, the search is called depth-first.

Pop

POP is a function on the stack. What is Stack? A stack is a set of data storage locations in such a way that the most recently stored item is the first to be retrieved. Accessing the content while removing it from the top of the stack is known as POP operation. In

one implementation of this function, the data item is not actually removed, but instead placed lower on the stack. How does POP function work?

Step 1 – Checks if the stack has an item or is empty.

Step 2 – If the stack has no item means it is empty then it throws an error.

Step 3 – If the stack has an item, the item is accessed.

Step 4 – Decreases the value of the next level of the stack by 1.

Step 5 – The POP operation is running successfully.

Mean

K-means is the most popular segmentation method. The K-means algorithm is an unsupervised, non-deterministic, numerical, iterative segmentation method. In this algorithm, each segment is represented by the mean value of the items in the segment. The algorithm consists of two separate phases:

1st Phase: a central point is chosen randomly, where the k value is predetermined.

2nd Phase: each item in the data set is associated with the nearest central point.

Euclidean distance is used to measure the distance between each item and a central point.

Item-item

Item-by-item collaborative filtering is an algorithm used in recommender systems based on the similarity between items, which is calculated using the rating users have given to items. It helps solve problems that user-based collaborative filtering suffers from, such as when the system has many items with fewer items rated.

Item-by-item models use per-item rather than per-user score dis-

tributions. When there are more users than items, each item tends to have more ratings than each user, so the average rating of an item usually does not change quickly. This leads to more stable score distributions in the model, so the model does not need to be rebuilt often. The main disadvantage is that item-to-item similarity tables must be calculated in advance, since updating the table when adding a new item is expensive.

User-user

The most commonly used algorithm follows the logic "people like you like this". Recommends items previously liked by similar users. The similarity between two users is calculated by the number of items they have in common in the dataset. This algorithm is efficient when the number of users is less than the number of items. The major disadvantage is that adding new users is expensive, as it requires updating all similarities between users.

FunkSVD

The FunkSVD algorithm ignores missing values from a table of items and users and finds a way to calculate the latent factors using only the values we know. To do this, the following steps are followed:

1. Two tables are created, one table of users based on the number of latent factors selected and one table of those latent factors per item. These tables are filled with random numbers. This step produces three (3) tables: a) the user-item table, b) the users and latent factors table filled with random values, and c) the latent factors and items table filled with random values.
2. Matrix a is searched for an existing value for a user-item pair
3. In matrix b, we get all the random values associated with the target user

4. Calculate the vector between the row and column found to make a prediction
5. The error is minimized using the GD (gradient descent) algorithm

BPR – Bayesian Personalized Ranking

This algorithm assumes that an observed interaction should be rated higher than unobserved interactions. BPR optimization is usually achieved with the stochastic gradient descent (SGD) algorithm. At each step, it first randomly selects an observed user-item interaction (u, i) and then selects an item j that user u has not interacted with before to yield (u, i, j) . Such a selection procedure j is known as negative sampling.

UserKNN + ItemKNN

The k Nearest Neighbors (kNN for short) algorithm is a “lazy” learning method. It has been considered as one of the simplest machine learning algorithms. The logic of kNN is that similar samples belonging to the same class have a high probability of being similar, while the basic idea of the algorithm is to first select the k nearest neighbors for each test sample. A disadvantage of the algorithm is that it is sensitive to the choice of the value of k . Although studies have focused on this topic for a long time, the selection of the k value in the kNN algorithm is still very difficult and controversial. The item-based kNN algorithm is a neighbor-based approach that belongs to the class of memory-based cooperative filtering algorithms. Instead of calculating the similarity of users, the algorithm calculates the similarity of items that the target user has rated (purchased or noticed) with the target item.

SVD++ ᳑ BMF (Biased Matrix Factorization)

SVD++ or BMF (Biased matrix factorization) algorithm refers to an optimized algorithm based on SVD. SVD is a matrix decomposition technique. Its role in collaborative filtering is mainly to analyze the users' preference for each factor and the extent to which the item contains each factor in the existing scoring scenario. Finally, the data is analyzed in turn to achieve the intended result.

Ratings in the user-item interaction panel are insights that have been leveraged in depth to generate recommendations. Due to the diversity in user-related score values for items, biased matrix factorization is used to improve score prediction. To overcome the sparsity of the scores, extra data is incorporated into the matrix factorization.

Random

The algorithm works by generating a random number r , within a specified range of numbers, and makes decisions based on the value of r . A randomized algorithm could help in a situation of doubt by flipping a coin or drawing a card from a deck in order to make a decision. Randomized algorithms are used when there is a time or memory constraint and an average solution is acceptable. Due to the possibility of the algorithm producing an incorrect result, an algorithm known as boosting is used to increase the probability of correctness by sacrificing execution time. Boosting works by repeating the randomized algorithm multiple times with different random samples of the input and comparing their results. It is common for randomized algorithms to boost only parts of the process, as overboosting can increase the execution time beyond the given constraints.

The algorithms that are most effective in managing bias in recommender systems are User-user/ UserKNN and Item-item/ ItemKNN. They work on the basis that items or users are similar, which is calculated based on the ratings users have given for the items. Therefore, they produce unbiased and fair recommendations. The depth-first search algorithm searches for the solution in depth, finds a solution, but has time and computational complexity. The pop function is a good solution, because it reorders the items based on their popularity, but this does not mean that the recommendations produced will be unbiased. The mean algorithm selects items based on the mean of the central point k and selects the items closest to that point. It does not take into account, however, the possibility that the data already contains bias and that the same data is recycled over and over again. So, it is not an unbiased algorithm.

The FunkSVD algorithm is a very good solution, but to produce recommendations it will need to predict values at random, so the results are not guaranteed to be unbiased. The negative sampling performed by the BPR (Bayesian Personalized Ranking) algorithm does not help with combating bias in recommender systems, because again it does not take into account that there may be bias in the data, so the higher ranking on the observed interactions does not matter roll. The SVD++ or BMF (Biased matrix factorization) algorithm selects random values to factorize the matrix of user scores for the items together with those provided by the users. Like other algorithms, this one does not take into account the fact that the data may contain bias, which is constantly being recycled. The Random algorithm randomly selects a solution to a problem. So, it doesn't help with the problem of bias in the data, which is the original goal.

Below is an explanation of the table with the metrics used to

Metrics in recommender systems against popularity bias

<i>Metrics/Systems</i>	Jabref/Sowiport (Collins, 2018)	Ekstrand (Ekstrand, 2018)	Feedback Loop (Mansoury M. A., 2020)	Mansoury (Abdollahpour, 2019)
CTR	✓	-	-	-
Chi-squared	✓	-	-	-
nDCG	-	✓	-	-
MRR	-	✓	-	-
KLD	-	-	✓	-
RF	-	-	-	✓
ΔGAP	-	-	-	✓

Table 10: Metrics used in recommender systems against popularity bias

measure the systems' performance. The CTR metric (Click Through Rates): is a metric that represents the ratio of the recommendations that users have clicked on in relation to the number of recommendations that they have received. The Chi-squared metric is a statistical hypothesis test used in matrix analysis when data sets are large. In essence, it is used to check if two vertical columns of a table are independent of each other, i.e., they do not affect the values of the table.

The NDCG (Normalized Discounted Cumulative Gain) metric is a metric that evaluates the classification quality of the generated recommendations. That is, it measures the profit that an item gets based on the position it has in the recommendation list. The MRR (Mean Reciprocal Rank) metric is a statistical metric that evaluates processes that generate lists of items for various questions. The rank of an item is the multiplicative inverse of the position of the first item. The average rank of the items indicates the average of the ranking results in the list. The KLD (Kullback-Leibler Divergence) metric: is a type of statistical distance: a measure of how two probability distributions differ from each other. The RF (Recommendation Frequency) metric measures how often an item is recommended to the user. The ΔGAP (Δ -Group Average Popularity) metric, i.e., the difference in the mean popularity of items (in the profiles) of

Explanatory table of elements in the GAP formula
g / N (user group)
$\phi(i)$ (popularity of an item)
P_u (item list in user's profile)
r_u (ratings in user's profile)

Table 11: Elements of GAP metric

users in a determined group g or in their lists of recommendations.

Example for the ΔGAP metric: The ΔGAP metric measures the difference in the amount of unwanted popularity for recommendations, which is imposed by the algorithms. If its value takes the value 0, then this means that the product providers in the recommender systems are fairly represented, that is, their products are recommended to users in the same way and at the same rate as the products of other providers. Its formula is as follows:

$$\Delta GAP(g) = \frac{GAP(g)r - GAP(g)p}{GAP(g)p}$$

To find the value of the above formula, the value of the GAP metric formula, which measures the average popularity of items on user profiles, should first be found. The formula for the GAP metric is as follows:

$$GAP(g) = \frac{\sum_{u \in g} \frac{\sum_{i \in P_u} \phi(i)}{|P_u|}}{|g|}$$

Here is an applied example of the GAP formula and then the ΔGAP formula:

For example, if we have the group of niche users, that is, those users who have special interests, the value of φ for this group will be 6.6, a value that results from the definition of φ , which is the ratio of how many times an item has been rated to the total number of users. So, since for this group we set a random number of items, about 8000 items, and the users in this group are about 1200, then the value is 6.6, because $8000/1200=6.6$. P_u is the list of items in the user's profile. We put a hypothetical value of 10 for this field. Finally, g is the user groups in the system, which are three (3). So, it will get the value 3.

Therefore, the only values that emerge from the data the authors give us are the value of g and the value of φ . The values for p_u and r_u are random and are the same for the purposes of applying the formula. Here are the values for the other two system user groups in the table below:

Table of value range for GAP and Δ GAP metrics	
P_u	10 (fixed value)
R_u	5 (fixed value)
$G \neq N$	3 (fixed value)
$\Phi(i)$	11,1/7,4/6,6

Table 12: Value range for GAP and Δ GAP metrics

Here is the applied example:

$$\begin{aligned}
GAP_{(g)} &= \sum u \in g \frac{\frac{\sum i \in pu \varphi(i)}{|pu|}}{|g|} \\
&\leftrightarrow \frac{\sum_{u=1}^3 \frac{\sum_{i=1}^{10} \varphi(i)}{|pu|}}{|g|} \\
&\leftrightarrow GAP_{(g)} = \frac{\frac{[(\varphi(1) + \varphi(2) + \varphi(3)) + (\varphi(1) + \varphi(2) + \varphi(3)) + (\varphi(1) + \varphi(2) + \varphi(3))]}{|pu|}}{|g|} \\
&\leftrightarrow \frac{\frac{[(6,6 + 11,1 + 7,4) + (6,6 + 11,1 + 7,4) + (6,6 + 11,1 + 7,4)]}{|10|}}{|3|} \\
&\leftrightarrow \frac{\frac{25,1 + 25,1 + 25,1}{10}}{3} \\
&\leftrightarrow \frac{\frac{75,3}{10}}{3} \\
&\leftrightarrow \frac{7,53}{3} \\
&\hookrightarrow 2,51
\end{aligned}$$

The GAP for the recommendations (GAP r) must then be found. We work with the same formula and where the Pu field was, we replace it with ru:

$$\begin{aligned}
GAP_{(g)r} &= \frac{\sum_{u \in N} \frac{\sum_{i \in ru} \varphi(i)}{|ru|}}{|N|} \\
&\leftrightarrow \frac{\sum_{u=1}^3 \frac{\sum_{i=1}^5 6,6}{|5|}}{|3|} \\
&\leftrightarrow \frac{\frac{[(\varphi(1) + \varphi(2) + \varphi(3)) + (\varphi(1) + \varphi(2) + \varphi(3)) + (\varphi(1) + \varphi(2) + \varphi(3))]}{|5|}}{|3|} \\
&\leftrightarrow \frac{\frac{25,1 + 25,1 + 25,1}{5}}{3} \\
&\leftrightarrow \frac{\frac{75,3}{5}}{3} \\
&\leftrightarrow \frac{15,06}{3} \\
&\hookrightarrow 5,02
\end{aligned}$$

Finally, we apply the formula for ΔGAP :

$$\frac{GAP_r - GAP_p}{GAP_p} = \frac{5,02 - 2,51}{2,51} = 1$$

We conclude that product providers in recommender systems are not fairly represented, since the ΔGAP metric took a value of 1.

3 Chapter 2 - Aspects of privacy in recommender systems

The abundance of products, services and information on the Internet has made recommender systems an integral part of the Internet. These are used in various situations and applications: from commercial websites to health applications. The benefits of personal recommendations are countless, for everyone involved. However, they bring to the spotlight some dangers, which may restrict the perception of those offering recommendations, such as the risk of privacy violations.

This risk lurks, because of the need for recommender systems to gather more and more personal data possible about the users subscribed to them. It is true that if a recommender system is to make recommendations, it is required to have some information about the users, in the form of a user model. This information can be used as a basis when there is need to generate recommendations, but in general, the quality of recommendations depends on the quantity, completeness and freshness of the latent data of the user model. On the other hand, the gravity of the risk regarding privacy and damage that exposure of user model data to third parties may cause depends on these factors. This is the so-called "privacy-personalization trade-off", which is observed when personalized recommendations are made (Awad and Krishnan 2006)(Chellappa and Sin 2005).

The privacy risks that are associated with personalization become worse when one deploys more sophisticated recommendation scripts. For instance, if one considers a recommender system that either refines its models for the users by extracting new characteristics and populates the information or combines various data sources for the user model. In such cases, the recommender system could

possibly reveal more information that was not accessible in the initial models for users, e.g., information that users could not have consented to get released for the purposes of recommendation. Exposure and access to this information by untrusted entities could lead to disastrous consequences.

3.1 Risks of Privacy in Recommender Systems

A lot of experts argue that people view their data as a product: they are eager to give away some of their personal data away for their own gain. Recommender systems comprise an example of that dynamic: They gather a vast variety of information of users and recommend much better services and recommendations to users (Cranor 2003) (Toch, Wang, and Cranor 2012). Data collected may include users' clicks or viewing behavior, information that is easy to predict, such as location and mood, social information, such as users' friends, family and colleagues, but also demographic parameters, such as age and employment (A. J. Jeckmans et al. 2013). To ensure responsible management of collected information by those collecting information, the Organization for Economic Co-operation and Development (OECD, 1980) has established a group of Fair Information Practices (FIPS):

1. Limitation of collection: Companies must collect data through legal and fair ways and with the consent of the user (where applicable).
2. Quality of Data: Data should be accurate, complete and relevant and should be kept up to date.
3. Determination of Purpose: Entities should determine the purposes of collection of data during the collection process.
4. Limitation of Use: Companies must not use or disclose infor-

mation for other purposes, except if they have consent from users and of course they have to do it lawfully.

5. Security Measures: Entities have to protect personal information against unauthorized access, against destruction, use, against conversion or exposure.

6. Availability/Candor: Companies have to let users should know what data they collect, who is in control of the data and why it is being used.

7. Individual Participation: Everyone has to be able to view their personal information that has been collected and delete, correct, complete or modify it.

8. Responsibility: Whoever collects the data must be in compliance with the above measures.

Privacy is disturbed when any of the above principles are violated. Given the need to collect large amounts of information and the innate ability to infer the personal interests of users from this data, recommender systems run the risk of violating the principles of Collection Limitation, Purpose Identification, Use Limitation, and Safeguards. In this direction, privacy risks can be categorized in two dimensions: when the violation of privacy is due to direct access to already existing data (violation of the principles of Collection Limitation and Limitation of Use) or because conclusions are drawn about new information (violation of the principle of Purpose Determination) and who this entity attempts to find information for users. There are considered to be three kinds of opponents:

1. The recommendation system interplays with the user, but may work in an inconsistent manner when it comes to the expectation of a user of privacy (breach of the principles of Limitation of Collection and Limitation of Use)

2. System users who cannot access the data of other users, but may take advantage of the facilities of the system to find data of the user they have set as a target (breach of the principle of Security Measures)

3. Some external entities who do not use the system, but might attempt to get access to the data that is held by the system or intercept with the interplay between the system and the users to intercept information (violation of the principle of Protection Measures, but this is different type of security measures).

3.2 Risks Caused by the Recommendation System

Recommender systems are usually based on a central entity, which has access to users' personal information in order to personalize services. However, the way the data is available, if it gets combined with commercial incentives, might result in violation of users' privacy expectations, even when the use of information is consistent with the provider's policy about privacy (Dhar, 2011). Direct access to data of users might put users at danger, including the following ways:

1. Unsolicited Information Collection: Since data storage capabilities are very affordable, online services get tempted to collect a lot of information about a user, either because it might be useful sometime in the future or because they could make money from it. However, collecting information that is not considered necessary in order to provide services can dissolve expectations about privacy. For instance, in a study designed to imprint the expectations users have about the sensitive information used by various applications, the Pandora Internet Radio application was one of the applications that got differentiated for using unexpected resources, because it

has access to the list of mobile phone contacts. Generally, users are suspicious of “environmental monitoring”, because third parties can make unwanted speculations about their data (Knijnenburg, 2013).

2. Sharing the data with third entities: recommender systems are motivated to share data of users with third entities. For instance:

- entities that gain access to data like that might want to share it so as to cooperate with a community of researchers, as in the case of AOL, which released anonymous searches, and the case of the Netflix Award.

- Some companies have to give data away to third entities in order to assign some parts of their business to other businesses. Nowadays, a lot of companies offer "recommendations as a service". The third entity gets records of user profiles and interactions from a site, works on them, creates recommendations and sends them back to the website. While a user's profile file might get anonymized before it gets sent, the third party now has a copy of the user's profile. Even if the user deletes their profile from the website, no one can guarantee that the third party has deleted the profile.

- Eventually, service providers might get tempted to sell personal information to brokers of data, as this has been proven to be a profitable activity. Moreover, information may change hands after a company is acquired or after an administrator sells off databases of businesses that have gone bankrupt. The important preoccupations of users whenever they give their information online are the spread of information from profiles which might be associated with a user (Ackerman, 1999) (Krishnamurthy, 2009). Even if the data controller takes precautions and anonymizes the data before publishing it to ensure user privacy, de-anonymization attacks are imminent to the data exposed.

3. Self-appointed access by employees: While the recommenda-

tion system can take providence to make sure that data of users is under control, it is possible that some of the employees might abuse the rights they have to search for data of people they know. Employees may, also, be enticed into stealing data from famous people (celebrities) out of curiosity or to gain money. This risk is inherent in every system containing such data and can be moderated by ensuring proper audit access and mechanisms for control.

3.3 Risks from conclusions drawn for new data

Complex management of the data collected or processed by the recommender system could lead to additional privacy risks, because conclusions are drawn on new data, often without the user's consent or absence:

1. Sensitive information exposure: Recent research has demonstrated the power that machine learning techniques have when it comes to revealing sensitive and personal data that include personality traits. Such guesses can be dangerous even if they are wrong, especially if they are made in the light of risk or bias.

2. Targeted advertising: In this field, aggregated information is used to find out interests of the users and choose specific ads which can lead to conversion. Ads that have targets might expose sensitive or degrading information – a good example of such ads is the case of a father who learned that their teenage daughter had been pregnant when Target began sending coupons to her for clothes and cribs for babies.

3. Discrimination: Recent work has shown evidence that online price discrimination exists and is aided by personal data. Users may perceive this as a misuse of their data and that it goes beyond the purposes identified for collection of data.

Guesswork attacks takes advantage of various aspects of the data of users to obtain sensitive data. These attacks are usually based on correlations made from data from other users, but can be exploited in a number of ways. For example, an entity can rely only on data from other users of the system (Weinsberg, 2012), form semantic relationships between various features (Chaabane, 2012), interlink data with other sources to extract new and more associations (Kosinski, 2013) or exploit the structure of social relations (Zheleva, 2009).

Demographic information such as gender, age, nationality or political affiliation could get inferred from information that got disclosed in recommender systems (Weinsberg, 2012). Many classifiers were trained with the use of the user data and made accurate inferences about the demographics of users that had revealed such data. Experiments carried out on the Flixster and the Movielens sets of data showcased the efficiency of this approach. Actually, simply watching a movie (regardless of rating for the movie) carries much information, meaning that classifiers that were trained on binary data (for example, whether someone saw the movie or not) performed only a little bit worse than those which were trained on integrated data.

Chaabane et al., used the ontological version of the Wikipedia page to find semantic relationships between interests of users and showed how seemingly innocuous interests such as interests around music, can divulge sensitive information about users (Chaabane, 2012), Weinsberg et al., took advantage of structured data (Weinsberg, 2012). If someone assumes that users with similar interests are similar in other domains as well, it is possible to make a guess about a private characteristic of a user based on public characteristics of

other users. The authors searched public Facebook user profiles and used users' exposed public interests in music to guess their gender, relationship status, age, and ethnicity.

Researchers conducted an extensive study that correlated users' Facebook likes with a set of sensitive characteristics, that included sexual orientation, ethnicity, religious and political views, and personality features using machine learning techniques. The authors predicted these sensitive features and achieved admirable results. For instance, the model could discern gay and straight men 88 % of the time, African Americans from Caucasian Americans 95 % of the time, and Democrats from Republicans 85 % of the time. Although some likes were associated with the trait under investigation (for example page likes related to homosexuality), some of the associations that were discovered didn't have any obvious relationships (Kosinski, 2013).

The guessing problem is exasperated in online social networks, where friendship links and group memberships can be used to extract personal information. Some researchers have taken into account that connected items in a social network might be correlated, for example that some people who are friends online might have similar features. They suggested several induction attacks that take advantage of the network structure to predict personal characteristics. The authors came to the conclusion that the predictions were based on the datasets with the use of data from Flickr, Facebook, BibSonomy and Dogster. For instance, methods based on links didn't perform well, because there was no possible relation between the inferred features and friends. On the other hand, group membership helped with inferring characteristics of users and many of the memberships allowed the prediction of user characteristics accurately. It is worth noting that while users can control which features can be

publicized, several social networks (such as Facebook and Flickr) do not really allow users to have control about the visibility of group membership (Zheleva, 2009).

3.4 Risks Other Users Impose

Recommender systems use data collected from countless users, so they permit them to collect personal data about one another, even when data like this remains private. This problem is mostly obvious when users use the same account on the same device or the same service: recommendations about that account could be drawn from the activities of both users, thus recommendations generated for a user give feedback on the activity of other users.

A more difficult situation occurs when the results of a recommender system divulge sensitive data about users of the system that have no relation between them. This mishap is particularly evident in recommender systems that use collaborative filtering algorithms, where of course these systems tailor recommendations for every user based on information collected from other users. The recommendations and explanations provided for them can expose information from users who rate items in completely different areas. Recommendations give the permission to an entity to infer relationships between items. For instance, given a specific item, an entity could make a fake account and give ratings about an item to find the smallest group of items that could lead to a suggestion for the item targeted. It is implied that there is a group of users who have given ratings about these items and the item that has been targeted. This user group is possibly going to be small when the assets belong to different domains, so it is going to be easier to target these users for attacks related to their privacy. For instance, the disclosed

connections could be combined with extra sources of information to reveal the identity of those users and more personal information (Ramakrishnan, 2001).

Calandrino et al., suggested a stronger attack that takes advantage of the public results of systems using item-based collaborative filtering algorithms (Calandrino, 2011). The public results of such recommender systems usually entail lists of similarity of items or correlations between them. For instance, Amazon provides “people who bought that product also have bought...” lists, Hunch offers the entire item-by-item covariance table and Last.fm gives lists of similarities of items. When an adversary observes changes in these scores over time, they could surmise a target user’s private transactions, given background knowledge of some items the user had rated. In a recommender system using item-based collaborative filtering algorithms, when a user purchases an item, this makes the item more similar to other items in their history of transactions. Therefore, the adversary could trace the similarity lists of items known to be related to the user and extract new items in the lists.

Additionally to the passive attack presented by some researchers, they, also, described an active “subversive” attack, which targets recommender systems using neighbor-based collaborative filtering algorithms (Calandrino, 2011). Based on prior knowledge about items that were rated by the user, the entity makes false users that are similar to the user in target and they will identify them as the user’s neighbors, as well as each other. A recommender system using neighbor-based collaborative filtering is going to possibly provide recommendations to false users based on other fictitious users and the user in target. This gives room for the isolation of the target user’s information, because any suggested item that doesn’t appear in the fictitious profiles is very likely to come from from the target

user.

3.5 Risks originating from external entities

Recommender systems control sharing and misuse of information and therefore may be propagated through regulation or divulged to get user consent. Conversely, some scenarios might lead to inadvertent disclosure of data. One of the risks come from hackers' illegal access to data (e.g., because of not sufficient security measures), resulting in theft of data. Another risk comes from law enforcement agencies' subpoenas. Although accessing data in this way is legal, it is often done without the knowledge of the user or in many cases, without the knowledge of the service provider.

Third entities may gain access to personal data collected from recommendation systems after they get anonymized to protect privacy. However, even the data is transformed into anonymized form, this data constitutes a privacy risk because it is possible to get de-anonymized. Some researchers have pointed that it is difficult to guarantee that transactions and preference records common to recommender systems will be anonymized (Narayanan, 2008). Generally, the sparsity of big multidimensional datasets ensures that a file will not be similar to many other files in the set of files, allowing it to be isolated and re-identified with little information. The attacker will be an adversary who knows a (possibly inaccurate) subset of the attributes of the target user, for example, items the user has rated, the ratings that the user assigned to items, or even the timeline these were made. Algorithms using de-anonymization calculate the how similar every file is in the anonymized set to pre-existing information. Because of the scarcity of the records of the transactions and the records regarding preferences, these algorithms

are imprecise and uncertain, but also have some level of disturbance to the issued records. The authors hypothesized that the noise required to overcome this de-anonymization approach could render collaborative filtering useless.

The amount in which this attack was effective was demonstrated with the use of the Netflix dataset, which contained anonymized ratings of five hundred thousand Netflix subscribers. The authors discovered that if we know ratings for eight movies (of which two may be wrong ratings) and the dates of the ratings vary with an error margin of one and fourteen days, we can re-identify 99 % of the files in the file set. Even if we do not know the dates on which the ratings were submitted, information on a few of the items rated might be enough. For instance, 84 % of files can be found again if the attacker has knowledge of six of the eight rated movies among the five hundred movies that have been more frequently rated. One can find this knowledge from most users, for example, by watching them willingly disclose information on social networks or IMDB. Anonymized files might not include sensitive data.

However, even in cases such as the above, re-identification contains risks: any information which can lead to a person could be used in successive attacks and give more advantages, which an attacker could use to de-anonymize more information. A collection of such information could lead to a "disaster database", which could combine digital traces taken from various sources, exposing a complex picture of various individuals' activities.

Netflix's second challenge was cancelled because of a out of court lawsuit about the re-identification of the Netflix dataset (Buley, 2010). Until now, the release of de-anonymized datasets securely for research purposes is an ongoing problem. In cases like this, "the reason data gets published is to enhance calculations on data which

were not seen before during publishing and are more complex than the computations that are known to us in a way that preserves privacy" (Narayanan, 2008)). Data speculation is therefore a privacy risk because it goes beyond users' initial privacy expectations.

3.6 Solutions on Privacy

The topic of the dangers of personal data that gets leaked through recommender systems leads to the side of the defender, that is, how user privacy can be protected by the recommender system without making the recommendations unfair. There are three categories of techniques that can help with the lack of privacy in recommender systems:

1. The first category comprises of architectures, platforms and standards that eliminate the threat of leakage of data. These include various protocols and certificates that guarantee users that the recommendation provider abides by privacy measures and protects the personal data of users. This minimizes access of external entities to user data. Also, this category entails distributed architectures, that minimize the single point of failure that tends to occur in centralized recommender systems.

2. The second category suggests algorithmic techniques that protect data. Here, some of the approaches stand out. Some of these include approaches about data modification. Others take advantage of proven privacy guarantees the differential privacy framework offers or implement tools that use cryptography to protect data.

3. The third category deals with legislation, policies and regulations. Governments and legislators could impose them or adopt them as self-regulatory labor practices. These can prevent services from managing, sharing or exchanging data.

3.7 System Design Architectures and Solutions

Trusted software to limit user connectivity and data dissemination

A recommender system can create user models by cross-referencing data from multiple sources. These models must not be retained after the recommendation process is terminated or disclosed to third groups, because this could constitute a serious danger to privacy of users. The recommendation system might claim that they do not store data or that they do not disclose data in order to reassure users with phrases like: "we do not reveal user profile data without consent from the users", "there is no connectivity between individual user sessions", "we limit the storage of data of users". Looking for solutions that preserve the privacy of recommendations, three ways to establish trust were identified:

Conclusion (Josang, 2007): Non-compliance could lead to negative feedback and feelings of users, which inhibits the use of the service from other users.

Certification (Stafford, 2001): A trusted third entity does a detailed technical audit, it analyzes the source code and runs tests so that it can identify all the qualities and properties the software has.

Trusted computing (Felten, 2003): An application should be able to identify a system's hardware and software, for example, if it has the ability to encode data in a decrypting way with a specific configuration (Cissee, 2007).

Portable profiles managed by users

Apart from the privacy risks stemming from guesswork and user profiles, social networking sites transformed into data repositories (Bonneau, 2009), with data of users' profiles getting locked or accessible partially through private interfaces. If the users could transfer information from a system to another, they could get better recommendations and more personalized services, solving the cold-start problem when they start using a new service. Moreover, they could permit access to specific profile data. Currently, however, this is not possible, because users cannot control their data this way yet.

At this point, attention is focused on the alternative architecture proposed by (Heitmann, 2010), which gives control to the user over their mobile profile information through semantic web techniques and through a control system that gives them access. With the use of this architecture, different services can share profile information and users can choose which profile parts they will disclose to the providers. The authors base their work model on three certificates/vocabularies:

- 1) Friend-of-a-Friend (Bojars, 2008): this is a data format that can store generalized data of user profiles, but also social friendships,
- 2) WebID (Canny, 2002): an SSL certificate, referring to the URI at which the profile information can be found, and
- 3) Web Access Control: a vocabulary that can control rights to access the resources.

Production of recommendations to the Client

Giving the attention to client devices and not only on the recommender system, allows the reduction of the load of information a recommender system accesses and retains, thereby reducing any privacy risk that could derive from exposure of a server to user data.

Many papers have suggested integrating the recommendation generation process as a peer-to-peer system, minimizing the role of a central service (Berkovsky S. E., 2006) (Lathia, 2007). However, user data could still be exposed to other users, whose interaction with the target user can be direct to create recommendations. Some researchers have managed this risk by proposing a similarity metric that is privacy-friendly based on interuser agreement, that is, the proportion to which sets agree with user ratings. This measure has the property of being evaluated they compare sets of scores with a third set, rather than being compared with sets relative to each other. Therefore, similarity between users can be assessed without sharing profiles (Lathia, 2007).

In a hybrid approach proposed by some researchers, every client has interactions with the central server to get recommendations, but can also trade data with users of the system to enhance privacy. Every user uses two profiles: an off-network profile located on the client, that is constantly kept up to date, and an on-network profile on the server, that is occasionally synchronized. So, every user's profile ends up being a mixture of each user's information. To keep the recommendations accurate, ratings given by similar users are favored over those not given by similar users (Shokri, 2009).

With computationally intensive recommendation algorithms and devices not suitable for heavy calculations, architectural approaches divide the work between a powerful backend device and a not so powerful user device. These models break recommendation generation into two parts:

- 1) modeling, for which requires the entire data set, and
- 2) recommendation generation, with which the recommendations are computed by the models. So, using this approach, generation of the recommendations can be a light task.

For instance, when it comes to item-based collaborative filtering, where the construction of an item-by-item similarity matrix is required, recommendations are produced using items similar to items that the user has previously consumed (Miller, 2004). In PocketLens, the authors began building a portable recommender system with collaborative filtering, where the calculation of similarity is distinct from the recommendation generation stage. Using homomorphic coding approaches, the backend generates an item-to-item similarity table and it does not have to decode individual records of purchases. A mobile client can find this table and generate recommendations locally. After they have integrated and evaluated several models, the authors found that the architecture that performs better could protect privacy of users without compromising the accuracy of recommendations.

Algorithmic solutions

Anonymization and pseudonyms

Anonymization and pseudonyms are algorithmic approaches, which camouflage users of recommender systems. At first, they were not well received. Anonymization techniques are disastrous for recommender systems, because they render the recommender system unable to identify the consumer, limiting its ability to collect data and make accurate recommendations. This topic remains unexplored to this day and there is little work in this field (Schafer, 2001).

A proposal for a personalization framework based on nicknames was created and gave rise to the concept of "personas". The framework implied that everyone has a bundle of abstract selves, i.e., entertainment, shopping, medicine, etc., and that these abstract entities are used when users interact with various services and sites. Multiple services are associated with each persona and expose only

the activities that persona has performed. Only one persona at a time is accessed through a service and cannot be linked to other personas, therefore, they cannot reveal additional information, while users handle their personas and set rights for access for different services and abstractions (Arlein, 2000).

User anonymization approaches in recommender systems typically involve simple solutions for de-identification. For instance, in the Netflix prize data, user IDs were replaced with random numbers. A major threat to this anonymization method lies in the high dimensionality and sparseness of the data (Narayanan, 2008), which is common in recommender system datasets. This sparsity can be exploited to undermine the anonymization and re-identify the files.

Masking

The application of data perturbation (or masking) techniques to recommender systems has inspired work outside of the recommender system field. The basic idea running through these works is that if anyone modifies some points in the user profile data, for example, if they add noise to the real information, this will have limited effects regarding the accuracy of the recommendations. However, if attackers or untrusted entities had access to profiles of users, they would just get the camouflaged profiles. This makes room for “truthful deniability” (Hancock, 2004): the entity cannot prove if a particular profile is real (Polat, 2003).

A random data perturbation technique was used by the authors to camouflage scores that were stored in user profiles. The data is modified by addition of random noise to the scores, so that no one can infer information about the scores. Because the recommendations are produced by adding the ratings of the users, the effect of data disturbance on the recommendations is minimal. Camouflaged

data was used in order to compare recommendations that were produced and those produced using real data, and it was demonstrated that the profiles that were perturbed could still produce accurate recommendations. How accurate the recommendations are is a topic inversely related to the amount of noise, but noise decreases depending on the number of items and users the recommender system has access to.

Parameswaran and Blough presented another version of data masking (Parameswaran, 2007). They suggested camouflaging auxiliary data related either to the items (e.g., metadata) or to the users (e.g. demographic data), which are used by the similarity calculation mechanism in conjunction with collaborative filtering. The evaluation demonstrated that the accuracy of recommendations is very small using obfuscation of auxiliary data, although they did not use obfuscation to prove that it can be used for user privacy.

Unfortunately, masking of data via noise addition cannot be applied to binary data, which is dominant in recommender systems, as systems constantly rely on binary behavior records (search records, shopping data, songs a user has listened to, etc.). In their paper, the authors applied a different approach, named "random response", to user profiles that have information on binary form. This approach selects which parts of the set with binary data to keep and which to discard randomly. Two versions of random responses were evaluated and accuracy was related to the amount of training data (Polat, 2006).

The implementation of random masking has moved over normal collaborative filtering. Yakut and Polat applied data masking to the Eigenstate-based version of collaborative filtering that works by reducing the dimensions of the matrix of scores using Principal Component Analysis (Yakut, 2007). Also, others implemented the

random response in an embedding in the collaborative filtering of a Naïve Bayes classifier. This work mainly focused on variations of noise parameters in order to maintain reasonable levels of privacy of users and accuracy of recommendations at the same time (Kaleli, 2007).

Data masking was applied by Basu et al., to the Slope-One recommender system, a version of feature-based collaborative filtering that can be scaled. Slope-One was noise-facilitated and capable of producing reasonably accurate recommendations despite data masking (Basu A. V., 2012). Data masking was, also, applied by Polat and Du to a recommender system that used the SVD algorithm with cooperative filtering, which decomposes the masked score matrix into three latent matrices. SVD recommendations have been found to perform well with random masking (Polat, 2005).

Renckes et al., applied data masking in a graph-based recommender system, in which users are the nodes and edges are used to calculate the similarity of the users (Renckes, 2012). This work verified Polat and Du’s findings on the effects of the availability of data regarding the accuracy of private recommendations and showed the trade-off between privacy and accuracy in practice. In short, the loss of privacy decreases as the level of obfuscation decreases, but the recommendation accuracy is also weakened, to the point where privacy and accuracy conflict with one another.

Kandappu et al. proposed an interactive obfuscation mechanism, in order for users to have more control over privacy and accuracy. Before the scores are shared in the system, the scores get masked. The recommender system is prompted by the engine to obtain predictions for ratings from self-provided items rated by the user, but not divulged to the system before the engine shares new ratings. Depending on the accuracy of the predictions, masking is adjusted

so that one can achieve privacy protection within a certain level of accuracy.

The attention of Berkovsky et al., was focused on applying data masking to various cooperative profile scores. They compared the effects of five data masking policies, which they applied to both "moderate" and "extreme" scores to find the accuracy of the generated recommendations. Masking in extreme scores had a greater effect on accuracy than masking in moderate scores. "Extreme" scores have more information in comparison to "moderate" ones, and adding noise to them results in making the accuracy of the recommendations weaker. However, "extreme" scores are considered to be more sensitive by users. If one does that, we can see the privacy-accuracy relationship differently, because if sensitive scores are masked, the accuracy of recommendations gets destroyed (Berkovsky S. K., 2012).

Differential privacy

Differential privacy acts as a technique that preserves privacy on the basis that after a computation no one should be able to guess anything about the records of the input. This can happen if we can require that the probability distribution for all outcomes that are possible does not change when files are added to or removed from the input. Thus, differential privacy shows the way to reduce guesswork about users' private data from system output. One common approach to achieve differential privacy is if anyone can utilize the Laplace mechanism, in which anyone can add noise to the computation that is beforehand carefully tuned and sampled from the Laplace distribution. Noise covers the effect that the difference in a particular file could have on the result of the calculation.

One of the first teams to study the differential privacy application in recommender systems was the team of McSherry and Mironov. The Laplace mechanism was used to take noisy sums and averages

from the input scores and generate a differential privacy version of the item-by-item covariance matrix. The noisy covariance matrix could then be used to produce differential privacy k-nearest neighbors and singular value decomposition (SVD) recommendations (McSherry, 2009).

Zhu et al., approached differentially private recommendations with cooperative "neighbor" filtering differently, specifically targeting the sybil attack described by Calandrino et al. They considered a differential privacy algorithm with k-nearest neighbor approximation, which works with two steps:

- 1) neighbor selection and
- 2) neighbor-based score prediction.

Another study explores privacy-preserving social recommendations based on a graph connecting users and items. A theoretical analysis of the problem was introduced by the authors and they deduced that anyone can achieve good recommendations if they weaken the privacy parameters, emphasizing that the privacy-accuracy relationship is included in methods based on differential privacy (Machanavajjhala, 2011).

Therefore, if we want to maintain privacy through multiple computations or data versions requires more noise to be introduced and leads to a decrease in accuracy. Although there are works studying effective differential privacy in continuous environments, there haven't been any studies across this field in recommender systems yet.

Solutions using cryptography

Solutions that use cryptography are the solution to use when user data has been exposed and we need to reduce privacy risks, such as intentional abuse (e.g., sharing information with third enti-

ties or making assumptions about sensitive data), as well as when disclosing data unintentionally (for example, data theft). Secure protocols using multiple computations allow recommendations to be correctly calculated. So, the user input is kept private. Secure computing produces the same recommendations as non-private protocols, unlike data obfuscation or differential privacy, but this leads to computational overhead, rendering these protocols primarily suitable for recommendations off the network.

Most of the study in this field is based on additive homomorphic coding schemes, such as the Paillier public-key cryptosystem. Many recommendation generation algorithms and architectures have used this property.

Cooperation between systems

Distributed algorithms are easily implemented between service providers, allowing cooperation between systems without divulging customer data to other systems, reducing privacy risks from sharing data with third entities. For instance, some scholars have explored the way a company can create recommendations basing their work on user data they own and other companies' data while maintaining user data private. They were based on homomorphic coding and safe comparison, absolute value as well as division protocols. The proposed a bilateral protocol that would be executed between two servers, which allows the production of predictions with user-to-user similarity and is evaluated with the use of the ratings given by users on each site (Jeckmans A. T., 2012).

Service provider

Several papers brought the problem of privacy risks in the interaction of the client with the server introducing a third party acting

as a provider of privacy to the forefront. The function of these solutions is based on the principle of "trust sharing", i.e., the information of the system is not owned completely by any entity in the system. The authors suggested using a framework for recommender systems that preserve privacy based on this element. Every agent is assigned to a "dealer" in the system, and the agent acts as a mediator in interacting with customers. The agent's public key encrypts profiles of customers so the agent can access them, but a merchant cannot. On the other hand, an image takes the place of the anonymized data, that only the merchant has knowledge of, so the agent is not in place to know which products the consumer purchases or rates. The agent knows which products are associated with a customer segment and a table of similarities in products, they manage them to create recommendations, and can keep them up to date based on user inputs, without having knowledge of what the products are (Aimeur, 2008).

There are more approaches to ensure recommendation computation apart from homomorphic coding. Nikolaenko et al., proposed a privacy preserving matrix factorization algorithm, in which the system makes profiles for the items without having knowledge about the user ratings. In the proposed protocol, a cryptographic service provider assists the recommender system. The service provider prepares a coded Yao circuit, whose job is to evaluate the profiles of items, only if it has knowledge of the inputs of the encrypted scores. The authors mention a low execution speed, and they consider the algorithm appropriate for truly large-scale data processing of large datasets (Nikolaenko, 2013).

The applicability of these solutions in a variety of environments and with various recommendation algorithms is shown in research conducted on cryptographic solutions for recommendations that pre-

serve privacy. In spite of that, these solutions consume considerable computing resources and time, but also storage space and communication overhead, which are obstacles for their implementation in online recommender systems.

Policies that offer solutions

As noted, a lot of states and countries actively regulate their consumer privacy, and many industry fields are adopting more privacy guidelines. There have been two proposals for the effects of laws for privacy and regulations about personalized systems up to 2006: one is the Privacy Bill of Rights of the United States and the review of the European Privacy Directive in 2012 (Kobsa A., 2007).

Huge significance on transparency and control is placed by both laws. For instance, the Bill of Rights for Privacy of the United States emphasizes that “there should be clear and simple choices about the collection of data of the consumers from companies, as well as about the use and disclosure of those personal data” and “companies must state completely clearly why they need the data and information about the way they will use it...”. The European Privacy Directive states that "personal data should be processed on the basis of the consent of the person concerned or on some other lawful basis".

The United States Privacy Statement further requires that consumers can access their personal data collected by companies and correct it if necessary. It also requires that data collection be targeted and limited to what is expected in the context for which the data was provided by the consumer. The European directive requires users to be able to access their data and be able to transmit the data from each service to another, but also to be able to delete it if they want.

The European Privacy Directive restricted using non-necessary

cookies and as a result, online advertising took a strong hit, especially in the European Union. This directive requires sites to clearly require permission from the users about non-essential cookies. This new directive has already been implemented in the laws of the Netherlands and the United Kingdom. However, most websites only provide users with two options so that they don't lose advertising money: they ask users to either quit or to accept the cookies and continue to the site. The constant display of cookie consent notices has created confusion for users, who accept them without knowing what they are consenting to.

Another way to privacy legislation is to auto-regulate the privacy laws is through trust seals. One of these seals is the TRUSTe seal or privacy standards such as P3P. TRUSTe seals are effective substitutes for legislation as demonstrated by Xu et al., because they have been proven to increase trust and reduce risk associated with privacy, while the P3P standard has been proven to increase trust, but not reduce risk (Xu, 2005). Auto-regulation has problems, too. Truste seals are partially effective, while various tests on commercial websites have shown that the seals have low conversion rates. P3P, in opposition, does not have high observability and suffers from complex user agents, which has led to users not adopting this approach.

3.8 How to build a privacy-respecting recommender system

Here are five (5) papers related to privacy and confidentiality in recommender systems, which shed light on the issue of privacy and explore ways in which various problems related to this issue can be overcome:

Description of the article “Adapting recommender systems to the new data privacy regulations”, written by Tejeda-Lorente A. et al, presented in the 17th International Conference on Intelligent Software Methodologies, Tools and Techniques in 2018

Background

Maintaining privacy in recommendation systems

Recommender systems provide users with personalized recommendations. This special service comes with infinite possibilities. Well-known examples where recommender systems have been implemented are all over in health, e-learning, e-commerce and other fields. As stated in the literature, recommender systems can be put into categories in a lot of ways. The categorization is based on the technique of generating recommendations, with two very important techniques, the collaborative filtering technique and the content-based recommendation generation technique. The content-based technique is based on how similar user profiles to item profiles are, while in the collaborative filtering technique the recommendations generated for users are based on their (historical – i.e. their behavior over time) similarities, their behavior and their interests and preferences compared to those of other users. The combination of all these different approaches into hybrid approaches, which are used to alleviate the weaknesses with the strengths of others (Tejeda-Lorente et al. 2018). Here is the structure of a recommendation system in the figure below:

Of the parts that make up a recommendation system, three parts stand out: The databases where information about the users and

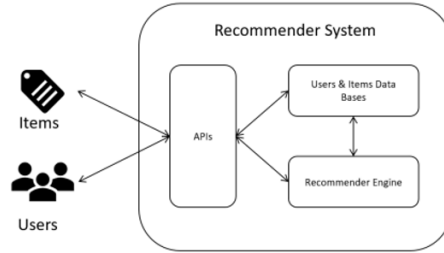


Figure 6: Recommender system structure

the elements of the system are stored, the interfaces for collecting or communicating the data from/and to the users and the elements and the recommendation engines, which create recommendations for users based on information generated and stored in the system. Different approaches can be used to estimate similarity between users. In recommender systems there is a majority of works in the literature that focus on privacy preservation using collaborative filtering. However, privacy preservation based on content-based recommendation generation technique is also covered in the literature.

The category of privacy-preserving collaborative filtering can be divided into two categories: • The first one aims to avoid the central server (for example, recommendation engines) to gain access to personal information, which is achieved with homomorphic encryption-based (HE) and secure multi-party computation (SMC). These techniques need high computing power, so they sacrifice scalability. • The second exchanges accuracy for anonymity. It introduces random jitter or masks the data before it sends the data to the central server.

Privacy Regulations

Data privacy law has been advancing along with technologies for collecting and processing personal information in recent years. A

resident has the right "to have his/her personal data protected", according to the Article 8 of the Law of Fundamental Rights of the European Union. The EU introduced the Data Protection Directive in 1995 and it aimed to ensure the privacy of residents from misuse or unnecessary collection of their data. However, lawmakers were prompted to create the new General Data Protection regulation, because of the vast majority of data created as social networks rised and because of the quest for personalization through big data in a digitized society. The regulation was implemented in May 2018. The General Data Protection Regulation (GDPR) brought major changes to the integration of privacy protection that all companies had to implement. All companies that handled data of EU residents had to obey the Regulation regardless of where they were based. Companies that collect, manage and use personal data have complied with data protection. There were changes introduced to the 1995 Directive (DIR95) by the GDPR, which focused on scopes of territory, new principles and made older principles and added conditions regarding parental consents clearer. There are two definitions of great importance in the GDPR, which directly affect recommendation systems: Personal data: The clause "personal information" is defined as "any information about a natural person with or without identity. An identified natural person is one who has identity, particularly referencing to something that identifies them, such as location data, a name, an online identifier an identification number, or one or more identifiers related to spiritual, economic, physiological, cultural, genetic, physical or social identity of a natural person". Profiling: "Profiling is defined as any form of automated processing of personal data to evaluate specific personal aspects about a natural person, namely the analysis or prediction of aspects concerning the natural person's performance at work, financial situation, health

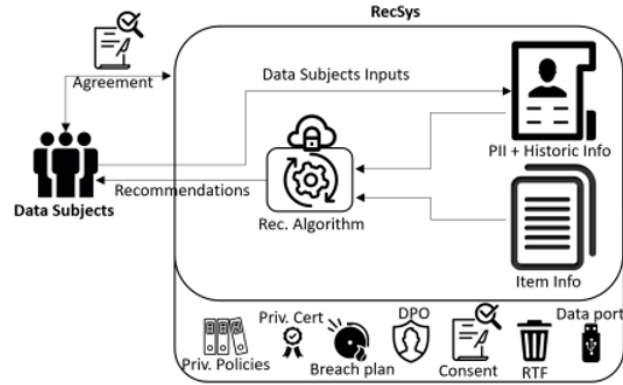


Figure 7: Privacy framework in recommender systems

his, his personal preferences, his interests, his reliability, his behavior, his location or his movements". The GDPR applied to personal data collected automatically in an indirect or direct manner. Companies are actually unaware of the GDPR and it has been a major challenge. Half of the companies were unaware about the new conditions and rules that the regulation would impose.

Privacy Compliance Guidelines

The obligations that the new regulations brought to companies forced them to review their privacy protection strategy and procedures. The GDPR incorporates the principle of accountability, i.e. the fact that a company is required to comply with the directive. For this reason, companies had to be provided with proper documentation of the policies developed and kept up to date and about the application of data protection certificates, seals or markings was recommended. The most important aspects regarding the design of a recommendation system when it comes to privacy management in the following framework are presented by the authors:

- Clear identification of data needs: The GDPR regulation is based on the minimization of data. This means limiting the processing of personal data required to a minimum. As a consequence, recommender system providers should inform data subjects (users) about this, i.e. explain specifically why they need the data they collect.
- Design of Privacy: Recommender system providers are required to abide by the GDPR and protect the rights of users and their privacy at each level.
- Data breach: Providers of recommendation systems must set up a process for notifying data protection authorities and users from breaches of data as early as possible.
- Data protection officer designation(DPO): As recommender system providers rely on frequent and tactical monitoring of users, they have to designate a data protection officer as the point of contact for all the issues of data protection.
- All users have to get clear information: Providers of recommendation systems are obligated to inform users in a transparent and comprehensible way about their rights, the legitimate interest used, the way the data was processed and under which legal basis was it executed . This means an end to the use of complex language within long text of terms and conditions to misguide users about the actual use of their information or rights.
- Consent for the use of personal data: The provider of recommendation systems should get consent from the user and should be in the position to prove it. The request should be distinguishable from all other information.
- The information should be given the right to be forgotten: Providers of recommendation systems have to delete personal information anytime the user requests it. If any third entity processes personal in-

formation, the managers have to establish a communication mechanism.

- The data must be portable: Providers of recommendation systems are obligated to give the user a copy of their personal information whenever they ask for it.

An argument commonly used when it comes to data processing regarding recommendation systems is legitimate interest, because it is the most flexible legal basis for processing information. The following table (table 13) contains the conditions for the legal and proper operation of a recommendation system:

Is the reason why the personal data is required explained?
Is the most important information explained in an understandable way?
Are users well informed about the purposes for which their data is used?
Is consent required separately from the normal terms and conditions?
Was the box checked previously?
Was the recommender system provider and, if data disclosed, the third party named?
Has the user been informed that they can withdraw their consent or refuse it without consequence/harm?
Consent was not a pre-requisite of the service
Was there an age verification process?
Are consent records kept correctly?
Are the contact information of the recommendation system provider and data protection officer provided?
If legitimate interest is used to communicate with users, is this explained?
Are individuals informed of the length of time data is stored?
Are users informed of their rights?
Are users informed about the process of creating profiles and generating recommendations?
If data is used by third parties, can they prove that they have consent?

Table 13: Conditions for the legal and correct operation of a recommender system

However, anyone should pay special attention to this point. The Information Commissioner’s Office (IOC) presents a guide to its use: someone can use data in ways they should expect it to be used and which have the least impact on privacy. Between all the aspects highlighted above, recommender systems are particularly influenced

by the design principle towards privacy. In [40], the authors state that companies should ensure an upfront integration of technical and organizational data protection measures. That is, appropriate measures that maintain protection taking into account integration costs and modern costs. With the GDPR, security measures must also be taken by processors of information. No clear guidance is provided when it comes to mechanisms about security, so the solution must be determined by the providers of the recommender systems. Based on the different techniques presented above, the authors suggest introducing the most appropriate measures that maintain protection along with the aspects already explained above. Each approach used to maintain privacy has its advantages and disadvantages. Therefore, since we depend on the peculiarities of the system, the most adequate of them should be chosen. As mentioned earlier, on the one hand, when accuracy is important, then the use of perturbation or masking techniques should be avoided, since these require that accuracy is reduced in order to provide privacy. On the other hand, they happen to be the most efficient techniques when it comes to scaling the solution when needed. Coding techniques address the performance problem, making it more difficult to achieve maximum performance in time-critical recommender systems, but provide more privacy for users. A possible solution for the coding process or the training of recommendation generation techniques in a controlled manner and one that preserves privacy is using third parties. The authors believe that if one follows the path that started with homomorphic coding, the main ways to follow when the goal is to strengthen the system's protection that affects performance as little as possible should be: the separation of knowledge and the anonymization or pseudo-anonymization. At the beginning of knowledge separation, the authors separate the personal data of the

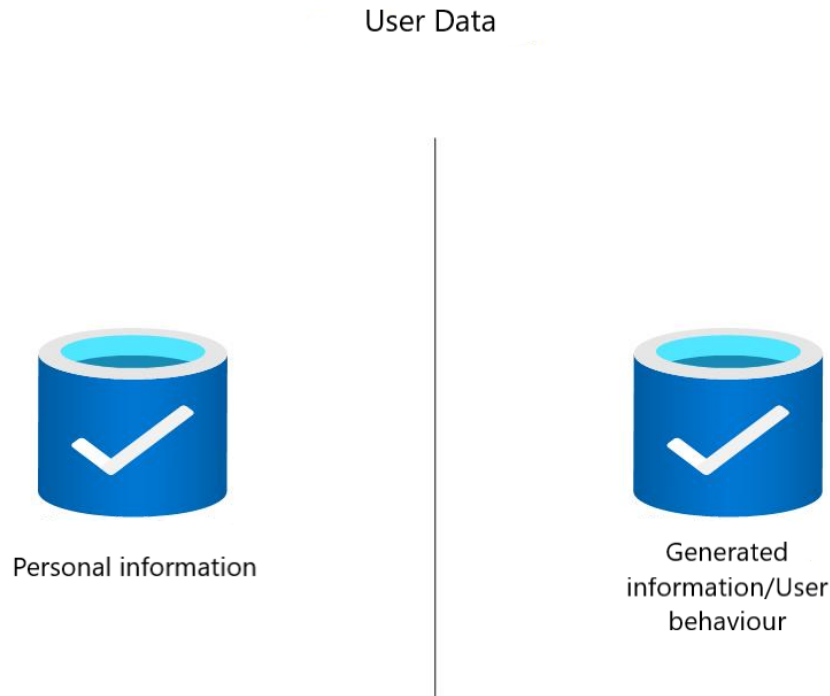


Figure 8: System architecture

most sensitive users from what they rate, their characteristics or maybe their behaviors from which the recommendation generation algorithms are “fed” either through trusted third parties or with a clear technical and organizational separation. That way, in the event of a data leak, the possibility of linking information to users or the possibility of using identification of users through information should be really hard.

With this as a base, the authors propose the architecture that was presented in figure 15, complete with these 2 rules:

The writers consider that following homomorphic encryption, the main rules to follow when strengthening the security level of a system in order to affect as less as possible its performance are: separation of knowledge and anonymity or pseudo-anonymity.

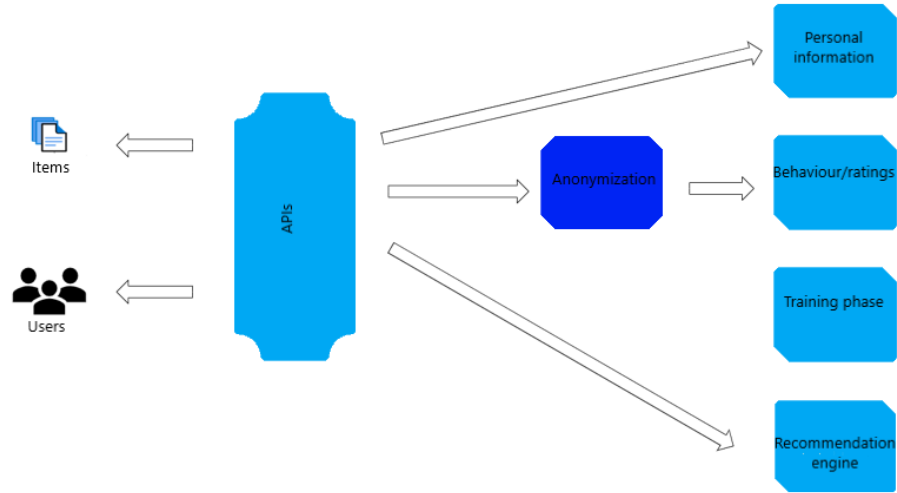


Figure 9: Model of a recommender system

When first attempting to separate knowledge, one has to separate the sensitive information of users from their ratings, their characteristics or their behaviours from which the recommendation generating algorithms feed from, either through trusted third entities or separating the information technically and organizationally. In this way, in case data gets leaked, the possibility of someone connecting information to users or in case of use of information to identify a user, it would be a very difficult task.

Moreover, when knowledge separation is done the right way, anonymity or pseudo-anonymity should be carried out as the norm each time personal information is stored or when it is used for reasons that have nothing to do with the final target of the existence of the system. When it comes to pseudo-anonymity, storing the keys is really restrictive. In order to achieve complete anonymity, we need to use a trusted third party.

The personal information required should be as less as possible and stored separately from ratings/behaviour that is generated from

users. Before storing information about ratings, it should be first anonymized. An discretionary step, would be the training of the algorithm that will be used for recommendation generation before with user data that has been placed before the generation of the recommendations in a way that ensures privacy.

Description of the article “Privacy preserving distributed machine learning with federated learning”, written by Chamikara M.A.P. et al., published in the Computer Communications Journal in 2021.

The problem to be solved is that data correlated with sectors, such as health, banking and other sectors, are sensitive and can be made public if their treatment is not effective. Federated learning is a method of distributed machine learning, which was developed for ensuring privacy. The motive behind this paper is the fact that a lot of attack methods take advantage of the vulnerable parts of machine learning models and of their servers, in order to gain access to sensitive personal data. The researchers introduce a new machine learning model, DISTPAB and prove that it provides with high accuracy, high effectiveness, high adaptability and high resilience to attacks. They, also, show that this system is an excellent solution for stopping the leakage of information, while preserving high utility for data (Chamikara et al. 2021). The writers showcase the three algorithms they have developed towards that scope. Algorithm 1 authorizes the preservation and modification of data to the distributed entities, leaving only the parameter generation to the central entity:

Algorithm 1: Centralized perturbation algorithm

$D \leftarrow$ original dataset
Input: $\sigma \leftarrow$ input noise standard deviation (default value=0.3)
Output: $D^\theta \leftarrow$ perturbed dataset

```
1  $\Phi = 0;$ 
2  $\theta_{optimal} = 0;$ 
3  $Rlf_{optimal} = 0;$ 
4 generate  $D^N$ ;
5 generate  $Cov(D^N)$ ;
6 generate  $TN^{noise}$ ;
7 for each  $ax$  in  $\{1, 2, \dots, n\}$  do
8   generate  $RF_{ax}$ ;
9   for each  $\theta_i$  do
10    generate  $M_i$  using Algorithm 4;
11     $\phi_{i,ax} = \min(\bar{1} + \text{diag}(M_i \times RF_{ax} \times Cov(D^N) \times RF_{ax} \times M_i^T) +$   

12       $\text{sum}((Cov(D^N) \times RF_{ax} \times M_i)^T)^T)$ 
13   $\phi_i = \min(\phi_{i,ax})$  where,  $ax \in \{1, 2, \dots, n\};$ 
14  $\Phi = \max(\phi_i)$  where,  $i \in \{1, 2, \dots, \theta\};$ 
15  $\theta_{optimal} = \theta_i$  at  $\Phi$ ;
16  $Rlf_{optimal} = ax$  at  $\Phi$ ;
17 generate  $M\theta$ ;
18 generate  $RF_{optimal}$ ;
19  $D^\theta = (M\theta \times TN^{noise} \times RF_{optimal} \times (D^N)^T)^T$ ;
20  $D^\theta = (\|D^\theta\| + \|\mathcal{N}(0, \sigma)\|) \bullet S_{\Sigma}$ ;
21  $D^\theta = D^\theta \bullet STDVEC + MEANVEC$ ;
22 randomly swap the tuples of  $D^\theta$ ;
```

Algorithm 2 shows the way a distributed entity works and algorithm 3 shows the way the central entity works:

Algorithm 2: Task of a distributed entity

Input: $D_i, M\theta, TN^{noise}, RF_{optimal}, STDVEC, MEANVEC$
Output: $D_i^\theta \leftarrow$ perturbed data partition

```
1 generate  $D_i^N$ ;
2 generate  $Cov(D_i^N)$ ;
3 generate  $Mean(D_i^N)$ ;
4 send Phase 1 data to the central entity;
5 receive Phase 2 data from the central entity;
6  $D_i^\theta = (M\theta \times TN^{noise} \times RF_{optimal} \times (D_i^N)^T)^T$ ;
7  $D_i^\theta = D_i^\theta + \mathcal{N}(0, \min(\sigma_{D_i^\theta}))$ ;
8  $D_i^\theta = D_i^\theta \bullet STDVEC + MEANVEC$ ;
9 randomly swap the tuples of  $D_i^\theta$ ;
```

Algorithm 3 shows how the central entity works. It calculates the global conversion parameters to send to the distributed entities. The authors assume that the distributed nodes are selected through a "handshake" mechanism, which confirms the validity and reliability of the distributed nodes. However, since the proposed algorithm does not share the original data with any third party before conversion, there is no data leakage during parameter communication:

Algorithm 3: Task of the central entity

$Cov(D_1), Cov(D_2), \dots, Cov(D_k) \leftarrow$ Covariance
matrices of the data partitions

Input: $Mean(D_1), Mean(D_2), \dots, Mean(D_k) \leftarrow$ Mean
vectors of the data partitions

Output: $M\theta, TN^{noise}, RF_{optimal}, STDVEC,$
 $MEANVEC$

```

1 receive Phase 1 data from the distributed entities;
2  $\Phi = 0$ ;
3  $\theta_{optimal} = 0$ ;
4 generate  $TN^{noise}$ ;
5  $Cov(D_1) = Cov(D_1)$ ;
6 for each covariance matrix,  $Cov(D_i)$  do
7   if  $i < k$  then
8      $Cov(D_{i+1}) = Cov(D_{i+1})$ ;
9      $Cov(D_{new}) = merge(Cov(D_i), Cov(D_{i+1}))$ , according to Eq.
      (15);
10     $Cov(D_i) = Cov(D_{new})$ ;
11  $Cov(D) = Cov(D_{new})$ ;
12  $STDVEC = \sqrt{diag(Cov(D))}$ ;
13  $MEANVEC = combine(Mean(D_1), Mean(D_2), \dots, Mean(D_k))$ ,
    using,  $MEANVEC = \frac{m_1 \times \sqrt{D_1} + m_2 \times \sqrt{D_2} + \dots + m_k \times \sqrt{D_k}}{m_1 + m_2 + \dots + m_k}$ ;
14 where,  $y_i = Mean(D_i)$ ;
15 for each  $ax$  in  $\{1, 2, \dots, n\}$  do
16   generate  $RF_{ax}$ ;
17   for each  $\theta_i$  do
18     generate  $M_i$  using Algorithm 4;
19      $\phi_{i,ax} = \min(\bar{1} + diag(M_i \times RF_{ax} \times Cov(D^N) \times RF_{ax} \times M_i^T) +$ 
       $sum((Cov(D^N) \times RF_{ax} \bullet M_i)^T))$ 
20 for each  $\theta_i$  do
21    $\phi_i = \min(\phi_{i,ax})$  where,  $ax \in \{1, 2, \dots, n\}$ ;
22  $\Phi = \max(\phi_i)$  where,  $i \in \{1, 2, \dots, \theta\}$ ;
23  $\theta_{optimal} = \theta_i$  at  $\Phi$ ;
24  $Rif_{optimal} = ax$  at  $\Phi$ ;
25 generate  $M\theta$ ;
26 generate  $RF_{optimal}$ ;
27 send Phase 2 data to the distributed entities;

```

Description of the article “An efficient blockchain-based privacy-preserving collaborative filtering architecture”, written by Uthaman P. Kavery, Ajeesh S. and Dr Smita C. Thomas, published in the International Research Journal of Engineering and Technology in 2021. Experimental data

The writers have used two very well-known datasets, MovieLens 100k and MovieLens 1M, so as to evaluate the quality of their methods. The ratings in each data set range from 1 to 5. Both datasets are scarce, because more than 90 % of the movie genres are empty. Initially, the methods B_KNN_GNA, B_AVG and B_KNN are put into practice and evaluated. Next, the researchers compare these techniques with a group of well known novel techniques. These are:

- 1) KNN (k nearest neighbors),
- 2) SGD (stochastic gradient descent),

- 3)ALS (alternating least square) and
- 4) NMF (non-negative matrix factorization) (Casino and Patsakis 2020).

Integration

Storing data directly in the blockchain is not effective. The reason why is that blockchains are designed to store transactions and not abstract data. Thus, someone could use systems like IPFS, Stori or TiesDB and put the identifying hash in the blockchain. This choice allows someone to store abstract data to the database effectively.

Effectiveness analysis

The authors have adapted the LSH superbit embedding in collaborative filtering methods as an efficient similarity calculation module. The number of stages was always 1.

Calculation of recommendations

The authors calculate the error between the values of the original data set and the predicted values using the Mean absolute error (MAE) metric, which is defined as:

MAE=

$$\frac{1}{n} \sum_{i=1}^n |p_i - r_i|$$

Where n is the number of elements, p_i is the predicted value for element i, and r_i is the actual value for i. The lower the MAE value, the better the forecast accuracy. Note that the recommendations are only calculated for the initial values given by each user. For each method, the authors conducted fivefold and tenfold tests of 100

times and then averaged the results. At the end, they choose the best configuration and compare their methods with the innovative techniques.

Description of the article “H2E: A privacy provisioning framework for collaborative filtering recommender system”, written by Ashraf Muhammad Usman et al., published in the Modern Education and Computer Science Journal in 2019.

The user can enter the system by identifying themselves. After that, there is a step of firebase identification. The user enters a code and their phone number in order to enter. The user chooses symptoms from a wide list of symptoms and the system tries to identify similarities between the symptoms and the illnesses using the identities of both. After that, the system calculates the similarities between the illnesses and the medication. After that, the system recommends the medication on the top of the list (with medication that has been rated) to the user, based on their interests or their preferences. The user can rate the medication and see which are the better rated medication, without affecting the system’s performance (Ashraf et al. 2019). Here is the recommended algorithm:

H2E Algorithm

Input: PNum $\leftarrow$ phone number

P $\leftarrow$ password

S $\leftarrow$ symptoms

Output: R $\leftarrow$ Recommendation

Declaration: T $\leftarrow$ timer, D $\leftarrow$ diseases, M $\leftarrow$ Medicine.

Key Generation:

Key generation involves one to four steps.

- (1) User enters personal information.
- (2) Check validation.
- (3) Code send to user mobile and add T.
- (4) Login.
- (5) Password Hash.

Recommendation Computing:

Recommendation generation involve five to nine steps.

- (6) Get S using Get () method.
 - (7) Find similarities between S and D.
 - (8) Count Sid Count ≥ 3 Did return.
 - (9) SELECT *FROM medicine WHERE Did LIKE Sid.
 - (10) Recommend M to user.
 - (11) User rates that M.
-

Here is the system's performance:

Description of the article “Towards privacy preserving recommender system with blockchains”, written by Al Omar Abdullah et al., presented in the International Conference on Dependability in Sensor, Cloud and Big Data Systems and Applications in 2019.

The writers in their platform use blockchain to block sharing private information. They use blockchain with permission, which provides control of access and finds the best recommendations when it comes to the user's interests, but also collaborative filtering. Through blockchain, the platform guarantees the security of the data, as users have access to blockchain to control and authorize the information sharing transaction (Omar et al. 2019).

Steps of the system:

Step 1: The user asks for recommendations.

Step 2: The identity generator receives the product type from the user and generates a unique identity corresponding to that user.

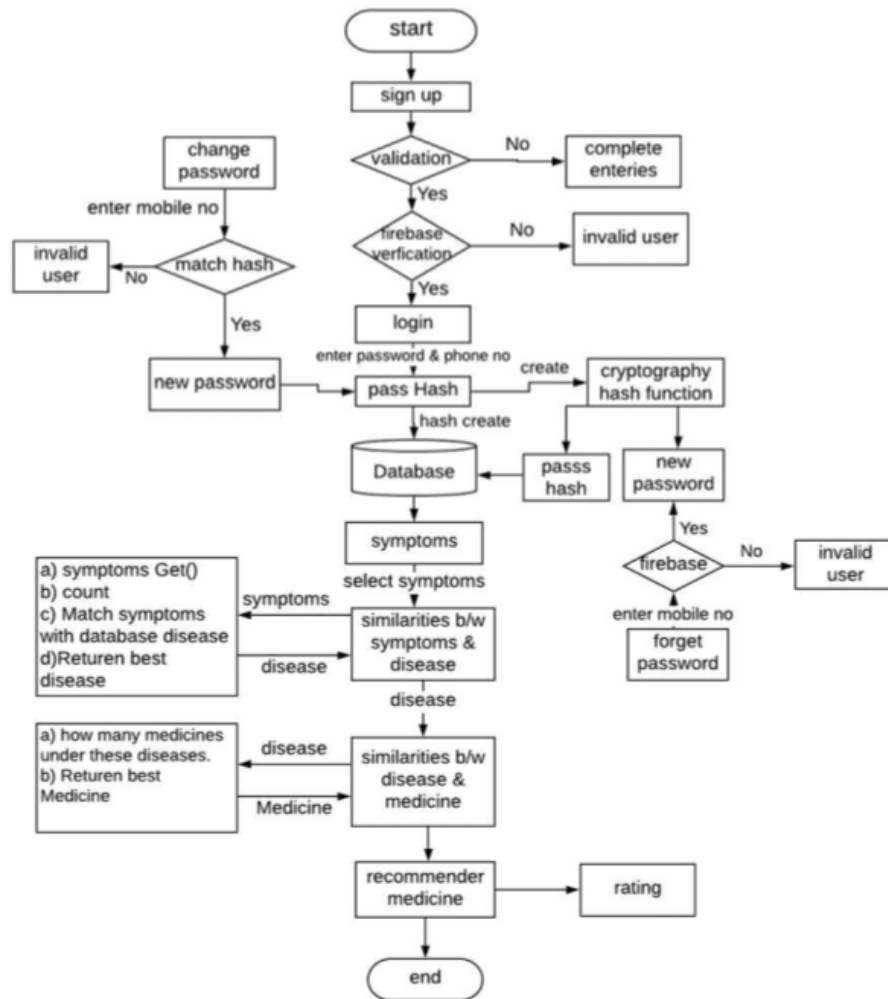


Figure 10: System architecture

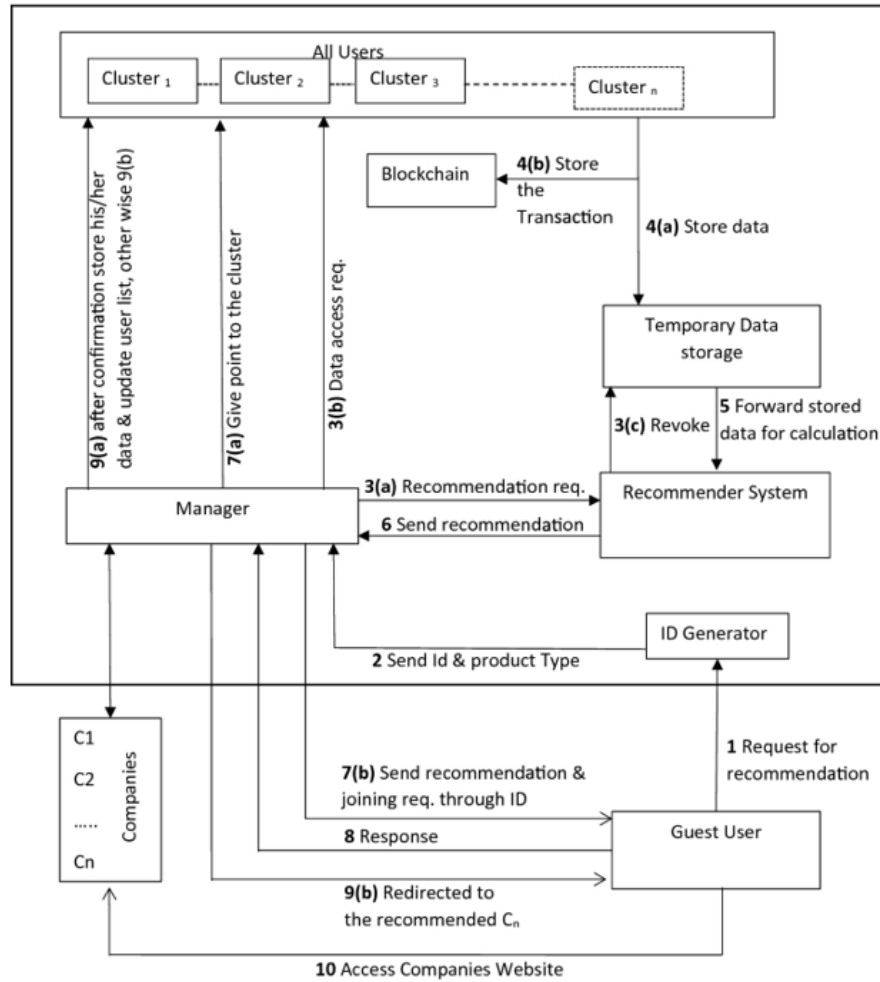


Figure 11: System architecture (Towards privacy preserving recommender system with blockchains)

After creating the identity, the generator sends the identity and the product type to the manager as a request for recommendations.

Step 3a: The manager stores the identity and sends a request for recommendations to the cloud.

Step 3b: The manager sends an access request for the data to the cloud.

Step 3c: The recommender system revokes the temporary data storage to give an answer about the data.

Step 4a: After the access, after permission from the user, the data is stored to the temporary data storage.

Step 4b: The transaction between the cloud and the temporary storage is stored in the blockchain.

Step 5: The temporary storage forwards the distributed data to the system to calculate the recommendations.

Step 6: The recommender system sends the recommendations to the manager.

Step 7a: When the manager gets the recommendations, it gives points to the cluster.

Step 7b: The manager forwards the recommendations to the user through their identity and requests from the user to connect to the system.

Step 8: The cluster sends the user's answer to the manager.

Step 9a: If the user is connected, the manager stores their data in the cluster and updates the user list.

Step 9b: The user receives the recommendations and is redirected to the page of a particular company or product.

Step 10: The user visits the suggested by the platform company page.

3.9 Categorization of privacy-respecting recommender systems

Finally, the above tasks are divided into categories based on their characteristics. These are the technologies they incorporate, the algorithms applied to them and the metrics for evaluating their performance.

<i>Recommender system technologies that preserve privacy</i>				
<i>Technologies/Systems</i>	DISTPAB (Chamikara, 2021)	Blockchain (Uthaman, 2021)	H2E (Ashraf, 2019)	Blockchain II (Al Omar)
<i>Federated learning</i>	✓	-	-	-
<i>Data perturbation</i>	✓	-	-	-
<i>Blockchain</i>	-	✓	-	✓
<i>LSH</i>	-	✓	-	-
<i>Cryptography</i>	-	-	✓	✓

Table 14: Technologies in recommender systems that promote privacy

In table 14, the DISTPAB system (Chamikara, 2021), uses federated learning, i.e., a technology that allows the use of distributed data, without sharing the original data. In addition, data perturbation technology is used, i.e., a technology that delegates the maintenance and alteration of data to distributed entities, leaving only the creation of data change parameters to the central entity. Distributed entities do not share the original data with any third party. Finally, the use of randomized expansion technology is observed, where the application of which causes the changing data to acquire randomness by adding noise to it. Also, it appears that the Blockchain system (Uthaman, 2021), uses blockchain (a technology characterized by a distributed series of data, which are grouped in blocks, i.e., in sections) and LSH (Locality sensitive hashing), i.e., a technique, which gives an abstract number of attributes to the items by giving them a signature, which allows achieving similarity between the items. In

addition, the 2HE system (Ashraf, 2019), uses encryption, a technology with which information can be encoded so that when it falls into the wrong hands, it is impossible to "read" and the Blockchain II system (Al Omar), uses blockchain and encryption.

Technologies

Federated Learning

Federated learning (FL) is a machine learning environment where multiple clients (e.g. mobile devices or entire organizations) jointly train a model under the orchestration of a central server (e.g. service provider), keeping the training data decentralized. FL incorporates the principles of data collection and minimization and can mitigate many of the systemic risks and privacy costs arising from traditional machine learning and data science approaches.

Data perturbation

A data obfuscation process can be described as follows: before data owners publish their data, they change the data in a specific way to mask sensitive information while preserving some specific property that is critical to building meaningful data mining models. Data obfuscation techniques must manage the trade-off between maintaining data privacy and maintaining data utility, as altered data reduces data utility.

Blockchain

The blockchain is a distributed database that maintains an ever-growing list of data records that are confirmed by the nodes participating in it. The data is recorded in a public file, which includes information about each completed transaction. Blockchain

is a decentralized solution, which does not require the involvement of third parties. Information about every transaction completed on the blockchain is shared and available to all nodes. This feature makes the system more transparent than centralized transactions involving a third party. In addition, nodes in the blockchain are all anonymous, which makes it more secure for other nodes to confirm transactions.

Locality sensitive hashing

The idea behind LSH (Locality sensitive hashing) is to use hash functions to map points into buckets, so that items that are closer to each other are more likely to be categorized into the same buckets than items that are further away. Finding the nearest neighbors is an important problem in many different applications. For small dimensions (<10), popular tree-based array structures are efficient, but for smaller dimensions, these structures suffer from the problem known as the curse of dimensionality. Instead of looking for exact results, one solution to the curse of dimensionality problem is to look for approximate results. In many applications where 100 % accuracy is not required, searching for results that are close enough is much faster than searching for exact results. The basic idea is to hash the points using several hash functions to make sure that for each function, the collision probability is higher for items that are close to each other than for those that are far apart. One can then determine the nearest neighbors by hashing the query point and the retrieval data stored in buckets containing that point.

Cryptography

Cryptography is the science of using mathematics to encrypt and decrypt data. Encryption enables sensitive information to be stored

or transmitted over unsecured networks so that it cannot be read by anyone but the intended recipient. A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in conjunction with a key – a word, number or phrase – to decrypt the plaintext. The same plaintext is encrypted into different ciphertext with different keys. The security of encrypted data depends entirely on two things: the strength of the cryptographic algorithm and the secrecy of the key. A cryptographic algorithm together with all the possible keys and all the protocols that enable it to work constitute a cryptosystem.

Regarding the technologies described above, the technology that really helps maintain user privacy is Blockchain, because users' sensitive data is divided into different blocks, so a malicious user cannot gain access to it. Its difference with other technologies is that it does not require the mediation of a third party, as in cryptography and federated learning. Also, it is a superior technology compared to data perturbation, since the latter presents problems related to data management (there is a balance problem between data privacy and their usefulness). Finally, it is a better method compared to LSH, because LSH is based on the KNN algorithm, so it assumes that items that are close to each other will also show similarities. This is not always correct. So, based on the above facts, systems using blockchain maintain privacy and excel over other systems.

The following table contains the categorization of the systems based on the algorithms applied to them.

In table 15, the DISTPAB system (Chamikara, 2021), uses the

Algorithms in Recommender systems that preserve privacy

<i>Algorithms/Systems</i>	DISTPAB (Chamikara, 2021)	Blockchain (Uthaman, 2021)	H2E (Ashraf, 2019)	Blockchain II (Al Omar)
<i>Bucketization</i>	-	✓	-	-
<i>Cryptography hash algorithm</i>	-	-	✓	-
<i>Schnorr signature algorithm</i>	-	-	-	✓
<i>Perturbation algorithm</i>	✓	-	-	-

Table 15: Algorithms that promote user privacy in recommender systems

perturbation algorithm, that is, this algorithm delegates the maintenance and change of data to the distributed entities, leaving only the creation of data change parameters to the central entity. Distributed entities do not share the original data with any third party. Also, it appears that the Blockchain system (Uthaman, 2021), uses the bucketization algorithm, where the algorithm takes the data inputs and processes them through a hash function, i.e., a hashing function. Through this function, each item/user rating, therefore each user is classified into a different segment or bucket. It is also observed that the H2E system (Ashraf, 2019) uses the cryptography hash algorithm: this algorithm is a classic example of a cryptographic algorithm and the Blockchain II (Al Omar) system, uses the Schnorr signature algorithm, where the user who wants to send a message has the ability to create signatures to authenticate their identity.

Algorithms

Bucketization

Bucketization is a process of grouping many records based on their sensitive values. Apparent sensitive values of the features are determined and ranked by frequencies in ascending order. After the

sorting is done, consecutive sensitive values are grouped into the same bin. Bins contain at least 1 distinct sensitive values that are retained after the binning process is complete. To add a sensitive value, the relevant bin is selected for processing. After all the sensitive values are binned, the larger bins are split into smaller ones. The split is order-preserving, i.e., the frequency of all values in the bin is less than that of all sensitive values in the bin. After their separation, all bins are sorted by the maximum frequency of their values in ascending order. At the end of the procedure, it returns a set of separated bins and at least 1 distinct values.

Cryptography hash algorithm

A hash function is a function that takes an arbitrary quantity for input and produces outputs of a fixed size. The standard hash function serves as a basis for studying cryptographic hash functions. In a highly secure system, data integrity is a critical aspect. With a cryptographic hashing function system users can create a message digest to detect unauthorized changes to files. It is especially important to verify data sources. Hash functions can be combined with other cryptographic methods. When hashing algorithms are combined with encryption, they produce special analyzes of messages called Message Authentication Codes.

Schnorr signature algorithm

A signature scheme is a cryptographic public key that guarantees the validity of messages. Until now, many schemes have been proposed, such as the El Gamal signature system, the Schnorr signature system, and the DSA system. The commonly accepted security concept for a signature scheme is the unforgeability of selected messages, which guarantees that even if an adversary obtains signa-

tures for messages, they cannot forge a valid signature to create a new message.

Perturbation algorithm

A data perturbation algorithm has been described above (see data perturbation).

In terms of algorithms, the one with the greatest potential is bucketization, because it divides data into buckets, making it impossible for a malicious user to gain access to them. The cryptography hash algorithm and the schnorr signature algorithm are based on cryptography, so this means that a third party can intercept and decrypt the messages, thus compromising the security of users' data. Also, the perturbation algorithm doesn't help much when it comes to data security. As mentioned before, it presents problems related to data management (there is a balance problem between data privacy and its usefulness). The system that is superior in terms of preserving user privacy is the Blockchain system described in the paper "An efficient blockchain-based privacy-preserving collaborative filtering architecture" (Uthaman, 2021).

In the following table (table 16) there is a categorization of the systems based on the metrics used by their creators to evaluate the performance of the systems.

The precision metric refers to the ability of a tool/model to classify only relevant data. Analogously, it refers to the ratio of true positive to true positive + false positive values. The recall metric is one that finds all relevant data within a data set. Analogously, it refers to the ratio of true positive to true positive + false negative values.

Metrics in recommender systems that promote user privacy

<i>Metrics/Systems</i>	DISTPAB (Chamikara, 2021)	Blockchain (Uthaman, 2021)	H2E (Ashraf, 2019)	Blockchain (Al Omar)	Π
<i>KNN</i>	✓	✓	-	-	
<i>SVM</i>	✓	-	-	-	
<i>Naïve Bayes</i>	✓	-	-	-	
<i>MLP</i>	✓	-	-	-	
<i>J48</i>	✓	-	-	-	
<i>SGD</i>	-	✓	-	-	
<i>ALS</i>	-	✓	-	-	
<i>NMF</i>	-	✓	-	-	
<i>MAE</i>	-	✓	-	-	
<i>Precision</i>	-	-	✓	-	
<i>Recall</i>	-	-	✓	-	

Table 16: Metrics used in recommender systems that promote user privacy

Metric KNN is an algorithm that works as follows: it uses proximity to make assumptions about the classification of data/items. It is commonly used as an algorithm for sorting items and works on the assumption that similar items/data can be found close to each other. The SGD (stochastic gradient descent) algorithm is an iterative algorithm, which starts from a point on a function until it reaches its lowest point. The goal of the ALS (Alternating least squares) algorithm is to find two matrices, e.g., M and N, whose values are almost the same as the values of the original matrices. When these values are found, the algorithm can predict whether a user will prefer a product or not.

The NMF (Non-Negative Matrix Factorization) algorithm is a method used to factorize a non-negative matrix X into 2 smaller matrices M and N. In this way, the size of the data is reduced, with the aim of better processing. The SVM (Support vector machine) algorithm aims to create a threshold that separates the layer into classes, to better classify the data into them. This limit is called the hyperplane. The Naïve Bayes algorithm is used for classification problems and works as follows: the data is transformed into frequency tables. It then creates the probability table by finding

the probabilities of the given items.

Finally, it uses Bayes' theorem to calculate the posterior probability. The assumption the algorithm makes is that each item is independent of the others. The presence of one characteristic does not affect the rest. The MLP (Multilayer Perceptron) algorithm is a neural network, where the mapping of inputs and outputs is non-linear. The J48 algorithm is an algorithm that produces decision trees based on training data, using the concept of information entropy. The MAE metric (Mean Absolute Error) is a metric that captures the average value of the magnitude of the error in a measurement.

3.10 A new fair recommendation system

In turn, I would like to propose a new system of recommendations, which will be fair and meet the needs of everyone within a platform or social media, both for users and service providers within them. This system combines the most promising technologies, techniques and algorithms in relation to the systems that have been examined in the previous chapters.

This new system will combine the well-known method of matrix factorization, but also Feedback loop simulation technology. The combination of these two methods would give a great advantage to a recommender system, since it will not only factorize the tables of items and users within the system, so that the system is fair and faster, since it will produce mostly fair recommendations, but the process of feedback loop simulation will already have preceded, that is, the system will have already predicted where problems will appear regarding how fair the recommendations it will produce for each user will be.

The system's algorithms

The best algorithms to make a recommender system fair are SVD, UserKNN, ItemKNN and ListRankMF. SVD reduces the size of tables for items and users. Then, by choosing the appropriate k , the UserKNN and ItemKNN algorithms will select the most relevant users and items respectively and finally, the ListRankMF algorithm, after taking the results of the two previous algorithms, i.e., the values of each table for all users and all items, it will generate the recommendation lists after first creating two tables for users and items and through them, get a feature vector, something like a user's rating for each item. This way it will produce a fair and accurate list of recommendations for each user every time.

3.11 A user privacy-oriented recommendation system

In this section, a recommendation system is proposed, which will preserve the privacy of users. This helps to hide users' personal data. No user wants everyone to know their personal information. According to the GDPR, all online service providers and not only, but also in general those who manage personal data of users and customers, are obliged not to disclose their sensitive personal data.

How could a recommendation system be created that takes this sensitive personal data into account? This is done with a technology called Blockchain. What is blockchain? It is a technology that works as follows: User data is stored in blocks, which include different items at a time. A block consists of these elements, the hash, i.e., something like a fingerprint, which is unique, and the hash of the previous block, with which it is linked to the previous block. This

hash, in case someone tries to change the data of the new block, changes and immediately makes the following blocks invalid. So, we know if something has changed on the blockchain.

However, since malicious users have found various ways to manage this peculiarity, another way has been invented to prevent easy access to the data of a blockchain, the so-called proof-of-work. According to this, the creation of a new block takes some time, that is, it delays its creation. If someone tries to break the proof-of-work, they will have to calculate the proof-of-work of all the remaining blocks of the blockchain. And the last reason why blockchain is the most secure method to create a recommender system, which will respect users' privacy is the fact that blockchain works decentralized. The data is no longer managed by a central entity, but is divided into a peer-to-peer network, meaning that anyone can participate in it. All nodes have a copy of the blockchain and can see if everything is in check. If everything is correct in a block, then there is agreement between all nodes. If something doesn't look right, it is rejected by all nodes. So, it becomes really laborious to hack user data.

The algorithms of the system

The recommender system, which will respect users' privacy, needs two algorithms to achieve this goal, the perturbation algorithm and the bucketization algorithm. The first puts noise into the data so that it falsifies it. If they fall into the wrong hands, malicious users will not be able to read them, since the original data is stored elsewhere. The second algorithm places the data in buckets, in small groups, i.e., it works decentralized, just like the blockchain. Therefore, even if some malicious users gain access to the data, they will

not have access to all of it, but to data that has been corrupted. So, it will be almost impossible to find user data and reveal information about them.

References

- Abdollahpour, Himan et al. (2019). “The unfairness of popularity bias in recommendation”. In: *arXiv preprint arXiv:1907.13286*.
- Ackerman, Mark S, Lorrie Faith Cranor, and Joseph Reagle (1999). “Privacy in e-commerce: examining user scenarios and privacy preferences”. In: *Proceedings of the 1st ACM Conference on Electronic Commerce*, pp. 1–8.
- Ameur, Esma et al. (2008). “Alambic: a privacy-preserving recommender system for electronic commerce”. In: *International Journal of Information Security* 7.5, pp. 307–334.
- Ashraf, Muhammad Usman et al. (2019). “H2E: A Privacy Provisioning Framework for Collaborative Filtering Recommender System”. In: *International Journal of Modern Education and Computer Science* 11.9, p. 1.
- Awad, Naveen Farag and Mayuram S Krishnan (2006). “The personalization privacy paradox: an empirical evaluation of information transparency and the willingness to be profiled online for personalization”. In: *MIS quarterly*, pp. 13–28.
- Baraglia, Ranieri et al. (2010). “Preserving privacy in web recommender systems”. In: *Privacy-Aware Knowledge Discovery, Novel Applications and New Techniques*. CRC Press, pp. 369–391.
- Basu, Anirban, Jaideep Vaidya, and Hiroaki Kikuchi (2012). “Perturbation based privacy preserving slope one predictors for collaborative filtering”. In: *IFIP International Conference on Trust Management*. Springer, pp. 17–35.
- Basu, Anirban, Jaideep Vaidya, Hiroaki Kikuchi, and Theo Dimitrakos (2013). “Privacy-preserving collaborative filtering on the cloud and practical implementation experiences”. In: *2013 IEEE Sixth International Conference on Cloud Computing*. IEEE, pp. 406–413.
- Bedi, Punam, Anjali Gautam, Chhavi Sharma, et al. (2014). “Using novelty score of unseen items to handle popularity bias in recommender systems”. In: *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, pp. 934–939.
- Berkovsky, Shlomo, Yaniv Eytani, et al. (2006). “Hierarchical neighborhood topology for privacy enhanced collaborative filtering”. In: *Proceedings of PEP06, CHI 2006 Workshop on Privacy-Enhanced Personalization, Montreal, Canada*, pp. 6–13.

- Berkovsky, Shlomo, Tsvi Kuflik, and Francesco Ricci (2012). “The impact of data obfuscation on the accuracy of collaborative filtering”. In: *Expert Systems with Applications* 39.5, pp. 5033–5042.
- Bobadilla, Jesús et al. (2020). “Deepfair: deep learning for improving fairness in recommender systems”. In: *arXiv preprint arXiv:2006.05255*.
- Bojārs, Uldis et al. (2008). “Social network and data portability using semantic web technologies”. In: *Proceedings of the BIS 2008 Workshop on Social Aspects of the Web, Innsbruck, Austria (May 2008)*.
- Bonneau, Joseph, Jonathan Anderson, and George Danezis (2009). “Prying data out of a social network”. In: *2009 international conference on advances in social network analysis and mining*. IEEE, pp. 249–254.
- Buley, Taylor (2010). “Netflix settles privacy lawsuit, cancels prize sequel”. In: *Forbes*, March 12.
- Calandrino, Joseph A et al. (2011). “" You might also like:" Privacy risks of collaborative filtering”. In: *2011 IEEE symposium on security and privacy*. IEEE, pp. 231–246.
- Canny, John (2002). “Collaborative filtering with privacy”. In: *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE, pp. 45–57.
- Casino, Fran and Constantinos Patsakis (2020). “An Efficient Blockchain-Based Privacy-Preserving Collaborative Filtering Architecture”. In: *IEEE Transactions on Engineering Management* 67, pp. 1501–1513. URL: <https://api.semanticscholar.org/CorpusID:208841496>.
- Chaabane, Abdelberi, Gergely Acs, Mohamed Ali Kaafar, et al. (2012). “You are what you like! information leakage through users’ interests”. In: *Proceedings of the 19th annual network & distributed system security symposium (NDSS)*. Citeseer.
- Chamikara, Mahawaga Arachchige Pathum et al. (2021). “Privacy preserving distributed machine learning with federated learning”. In: *Computer Communications* 171, pp. 112–125.
- Chellappa, Ramnath K and Raymond G Sin (2005). “Personalization versus privacy: An empirical examination of the online consumer’s dilemma”. In: *Information technology and management* 6, pp. 181–202.
- Chen, Terence et al. (2014). “On the effectiveness of obfuscation techniques in online social networks”. In: *Privacy Enhancing Technologies: 14th Interna-*

- tional Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings 14*. Springer, pp. 42–62.
- Cissée, Richard and Sahin Albayrak (2007). “An agent-based approach for privacy-preserving recommender systems”. In: *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 1–8.
- Collins, Andrew et al. (2018). “A study of position bias in digital library recommender systems”. In: *arXiv preprint arXiv:1802.06565*.
- Cranor, Lorrie Faith (2003). “I didn’t buy it for myself’privacy and ecommerce personalization”. In: *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pp. 111–117.
- Crnkovic, Ivica et al. (2001). “4th ICSE workshop on Component-Based software engineering: Component certification and system prediction”. In: *ACM SIGSOFT Software Engineering Notes* 26.6, pp. 33–40.
- Dhar, Vasant, Jessy Hsieh, and Arun Sundararajan (2011). “Comments on ‘Protecting Consumer Privacy in an Era of Rapid Change: A proposed Framework for Businesses and Policymakers’”. In: .
- Ekstrand, Michael D et al. (2018). “All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness”. In: *Conference on fairness, accountability and transparency*. PMLR, pp. 172–186.
- Felten, Edward W (2003). “Understanding trusted computing: will its benefits outweigh its drawbacks?”. In: *IEEE Security & Privacy* 1.3, pp. 60–62.
- Hancock, Jeffrey T, Jennifer Thom-Santelli, and Thompson Ritchie (2004). “Deception and design: The impact of communication technology on lying behavior”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 129–134.
- Heitmann, Benjamin et al. (2010). “An architecture for privacy-enabled user profile portability on the web of data”. In: *Proceedings of the 1st International workshop on Information Heterogeneity and Fusion in Recommender Systems*, pp. 16–23.
- Isaacman, Sibren et al. (2011). “Distributed rating prediction in user generated content streams”. In: *Proceedings of the fifth ACM conference on Recommender systems*, pp. 69–76.
- Jeckmans, Arjan, Qiang Tang, and Pieter Hartel (2012). “Privacy-preserving collaborative filtering based on horizontally partitioned dataset”. In: *2012*

- International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, pp. 439–446.
- Jeckmans, Arjan JP et al. (2013). “Privacy in recommender systems”. In: *Social media retrieval*, pp. 263–281.
- Jøsang, Audun, Roslan Ismail, and Colin Boyd (2007). “A survey of trust and reputation systems for online service provision”. In: *Decision support systems* 43.2, pp. 618–644.
- Kaleli, Cihan and Huseyin Polat (2007). “Providing private recommendations using naive Bayesian classifier”. In: *Advances in Intelligent Web Mastering: Proceedings of the 5th Atlantic Web Intelligence Conference–AWIC’2007, Fontainebleau, France, June 25–27, 2007*. Springer, pp. 168–173.
- Knijnenburg, Bart P and Alfred Kobsa (2013). “Making decisions about privacy: information disclosure in context-aware recommender systems”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 3.3, pp. 1–23.
- Kobsa, Alfred (2007). “Privacy-enhanced web personalization”. In: *The adaptive web: Methods and strategies of web personalization*. Springer, pp. 628–670.
- Kobsa, Alfred and Jörg Schreck (2003). “Privacy through pseudonymity in user-adaptive systems”. In: *ACM Transactions on Internet Technology (TOIT)* 3.2, pp. 149–183.
- Kosinski, Michal, David Stillwell, and Thore Graepel (2013). “Private traits and attributes are predictable from digital records of human behavior”. In: *Proceedings of the national academy of sciences* 110.15, pp. 5802–5805.
- Krishnamurthy, Balachander and Craig Wills (2009). “Privacy diffusion on the web: a longitudinal perspective”. In: *Proceedings of the 18th international conference on World wide web*, pp. 541–550.
- Lathia, Neal, Stephen Hailes, and Licia Capra (2007). “Private distributed collaborative filtering using estimated concordance measures”. In: *Proceedings of the 2007 ACM conference on Recommender systems*, pp. 1–8.
- Ma, Zhe and Qiang Dong (2021). “Alleviating the unfairness of recommendation by eliminating the conformity bias”. In: *Journal of Physics: Conference Series*. Vol. 2004. 1. IOP Publishing, p. 012008.
- Machanavajjhala, Ashwin, Aleksandra Korolova, and Atish Das Sarma (2011). “Personalized social recommendations-accurate or private?” In: *arXiv preprint arXiv:1105.4254*.

- Mansoury, Masoud, Himan Abdollahpouri, Mykola Pechenizkiy, et al. (2020). “Feedback loop and bias amplification in recommender systems”. In: *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 2145–2148.
- Mansoury, Masoud, Himan Abdollahpouri, Jessie Smith, et al. (2020). “Investigating potential factors associated with gender discrimination in collaborative recommender systems”. In: *arXiv preprint arXiv:2002.07786*.
- Markus, Robert M Arlein Ben Jai and Jakobsson Fabian Monrose Michael K Reiter (2000). “Privacy-Preserving Global Customization”. In: *EC’00: Proceedings of the 2nd ACM Conference on Electronic Commerce: Minneapolis, Minnesota, October 17-20, 2000*. Citeseer, p. 176.
- McSherry, Frank and Ilya Mironov (2009). “Differentially private recommender systems: Building privacy into the netflix prize contenders”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 627–636.
- Miller, Bradley N, Joseph A Konstan, and John Riedl (2004). “Pocketlens: Toward a personal recommender system”. In: *ACM Transactions on Information Systems (TOIS)* 22.3, pp. 437–476.
- Narayanan, Arvind and Vitaly Shmatikov (2008). “Robust de-anonymization of large sparse datasets”. In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, pp. 111–125.
- Nikolaenko, Valeria et al. (2013). “Privacy-preserving matrix factorization”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 801–812.
- Omar, Abdullah Al et al. (2019). “Towards privacy-preserving recommender system with blockchains”. In: *Dependability in Sensor, Cloud, and Big Data Systems and Applications: 5th International Conference, DependSys 2019, Guangzhou, China, November 12–15, 2019, Proceedings 5*. Springer, pp. 106–118.
- Parameswaran, Rupa and Douglas M Blough (2007). “Privacy preserving collaborative filtering using data obfuscation”. In: *2007 IEEE International Conference on Granular Computing (GRC 2007)*. IEEE, pp. 380–380.
- Polat, Huseyin and Wenliang Du (2003). “Privacy-preserving collaborative filtering using randomized perturbation techniques”. In: *Third IEEE international conference on data mining*. IEEE, pp. 625–628.

- Polat, Huseyin and Wenliang Du (2005). “SVD-based collaborative filtering with privacy”. In: *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 791–795.
- (2006). “Achieving private recommendations using randomized response techniques”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pp. 637–646.
- Put, Andreas et al. (2014). “inshopnito: An advanced yet privacy-friendly mobile shopping application”. In: *2014 IEEE World Congress on Services*. IEEE, pp. 129–136.
- Ramakrishnan, Naren et al. (Dec. 2001). “Privacy risks in recommender systems”. In: *Internet Computing, IEEE* 5, pp. 54–63. DOI: 10.1109/4236.968832.
- Regard, Having (1980). “Recommendation of the Council concerning Guidelines governing the Protection of Privacy and Transborder Flows of Personal Data (2013)”. In.
- Renckes, Sahin, Huseyin Polat, and Yusuf Oysal (2012). “A new hybrid recommendation algorithm with privacy”. In: *Expert Systems* 29.1, pp. 39–55.
- Riboni, Daniele and Claudio Bettini (2012). “Private context-aware recommendation of points of interest: An initial investigation”. In: *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*. IEEE, pp. 584–589.
- Schafer, J Ben, Joseph A Konstan, and John Riedl (2001). “E-commerce recommendation applications”. In: *Data mining and knowledge discovery* 5, pp. 115–153.
- Seymen, Sinan, Himan Abdollahpouri, and Edward C Malthouse (2021). “A Unified Optimization Toolbox for Solving Popularity Bias, Fairness, and Diversity in Recommender Systems.” In: *MORS@ RecSys*.
- Shokri, Reza et al. (2009). “Preserving privacy in collaborative filtering through distributed aggregation of offline profiles”. In: *Proceedings of the third ACM conference on Recommender systems*, pp. 157–164.
- Sonboli, Nasim and Robin Burke (2019). “Localized fairness in recommender systems”. In: *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, pp. 295–300.
- Suguri, Hiroki (1999). “A standardization effort for agent technologies: The foundation for intelligent physical agents and its activities”. In: *Proceedings of the*

- 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers.* IEEE, 10–pp.
- Tejeda-Lorente, Álvaro et al. (2018). “Adapting recommender systems to the new data privacy regulations”. In: *New Trends in Intelligent Software Methodologies, Tools and Techniques*. IOS Press, pp. 373–385.
- Toch, Eran, Yang Wang, and Lorrie Faith Cranor (2012). “Personalization and privacy: a survey of privacy risks and remedies in personalization-based systems”. In: *User Modeling and User-Adapted Interaction 22*, pp. 203–220.
- Tupe, Umakant L and RB Joshi (2014). “Privacy Preserving Dynamic Recommender System”. In: *International Journal of Computer Applications* 105.9.
- Vallet, David, Arik Friedman, and Shlomo Berkovsky (2014). “Matrix factorization without user data retention”. In: *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I 18*. Springer, pp. 569–580.
- Weinsberg, Udi et al. (2012). “BlurMe: Inferring and obfuscating user gender based on ratings”. In: *Proceedings of the sixth ACM conference on Recommender systems*, pp. 195–202.
- Xu, Heng, Hock-Hai Teo, and Bernard Tan (2005). “Predicting the adoption of location-based services: the role of trust and perceived privacy risk”. In.
- Yakut, Ibrahim and Huseyin Polat (2007). “Privacy-preserving eigentaste-based collaborative filtering”. In: *Advances in Information and Computer Security: Second International Workshop on Security, IWSEC 2007, Nara, Japan, October 29-31, 2007. Proceedings 2*. Springer, pp. 169–184.
- Zheleva, Elena and Lise Getoor (2009). “To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles”. In: *Proceedings of the 18th international conference on World wide web*, pp. 531–540.
- Zhu, Tianqing et al. (2014). “An effective privacy preserving algorithm for neighborhood-based collaborative filtering”. In: *Future Generation Computer Systems* 36, pp. 142–155.