Radboud University Nijmegen

Faculty of Science

Protecting Implicit Sensitive Information from Inference Attacks

Toward Gender Obfuscation in the User-Item Matrix

Master Thesis

Supervisor:
Martha Larson

Author:
Christopher Strucks

External Supervisor:
Manel Slokom

Second Reader:
Eelco Herder

August 2019

Abstract

Collecting as much data as possible is commonly seen as a necessary method for effective machine learning algorithms. However, privacy concerns rise if the data contains personal and possibly sensitive user information. While there are regulations, like the GDPR, to keep the sensitive information of the collected data protected, it is still possible to extract implicit user-demographic information such as a user's gender. A study by Weinsberg et al. (2012) has addressed this issue by removing implicit gender information from the user-item matrix in a movie ratings dataset; however their algorithm fails to take into account that simple data visualization can reveal that data obfuscation has been applied. Building on the work of Weinsberg et al. (2012), we thus present two improved binary gender obfuscation algorithms to counter this problem. This approach enables the resulting obfuscated datasets to be robust to gender inference attacks in such a way that an attacker can hardly determine if the dataset is obfuscated. Our findings provide important implications for and contribute toward methods to protect user privacy and facilitate secure data sharing in a multi-stakeholder environment.

Contents

1 Introduction **3**

2 Background & Related Work **6**
2.1 Recommender Systems 6
2.2 Recommendation Strategies 6
2.3 Evaluation of Recommendation Systems 8
2.4 Threat Model 8
2.5 Privacy in Recommender Systems 8
2.6 Gender in a User-Item Matrix 10
2.7 Realistic fake data 10

3 Experimental Setup **12**
3.1 Data 12
3.2 Re-implementing BlurMe 12

4 Part 1: Relevant Attributes for Gender Inference **15**
4.1 Data Saturation 15
4.2 Binary User-Item Matrix 15
4.3 Indicative movies 16
4.4 Genre as important attribute 19
4.5 Discussion - Relevant Attributes 21

5 Part 2: Data Obfuscation - Building a Better BlurMe **23**
5.1 Threat Model 23
5.2 The Issue with BlurMe 23
5.3 The Definition of BlurM(or)e 24
5.4 Comparison of BlurMe and BlurM(or)e 25
5.5 Recommendation performance 26
5.6 Generalizability 26
5.7 Discussion - Obfuscation 27

6 Part 3: Approaches for More Realistic Data **29**
6.1 General Threat model 29
6.2 Experiments - Real vs. Fake 30
6.3 Toward More Realistic Data 31
6.4 Quality of the Dataset 36
6.5 Generalizability 36
6.6 Discussion - Realistic data 38

7 Overall Discussion and Conclusion **39**

Appendix **46**

1 Introduction

Throughout recent years, the internet has become increasingly more intertwined with our personal lives, requiring us to provide sensitive data for personalized experiences. When creating online accounts this data is stored at respective websites and may be accessible to third parties without the user's consent. Users may be aware of this and subsequently avoid providing sensitive data. However, it is less likely that they are aware of the fact that personal characteristics can be inferred by analyzing their behavioral patterns on websites. In 2006, for instance, Netflix, Inc., the well-known online movie streaming platform, created an open competition (the "Netflix Prize" competition) to find an improvement on their rating prediction performance. They uploaded their dataset of user-item interactions, so that everyone that attempted the competition could use the same dataset. The users in the dataset were anonymized, but only 2 years later Narayanan and Shmatikov (2008) [33] showed that it is possible to de-anonymize the Netflix Prize dataset, revealing the identity and other private information such as political orientation of users. In addition, Weinsberg, Bhagat, Ioannidis and Taft (2012) [50] prove that e.g. movie-rating data contains hidden, possibly sensitive information, which can be inferred without the users' knowledge. As a result, raising privacy concerns put pressure on internet companies and demand measures that at least complicate inference of personal data. In their study, Weinsberg et al. (2012) proposed a data obfuscation mechanism by adding fictional data to the original dataset in order to obfuscate binary gender information of a user. Of course, data obfuscation comes at a cost of data quality and usability, which must be maintained for data sharing purposes. Hence, the goal of data obfuscation is to modify the data in such a way that the privacy of users is maximized while maintaining the quality of the dataset.

By means of simple data visualization - or more complex machine learning techniques - however, it is possible to identify successful data obfuscation. This is of course a vulnerability of the data obfuscation algorithm, which can be exploited by an attacker. If the attacker knows that a dataset is obfuscated, he or she could potentially reverse the gender obfuscation (by e.g. removing items from the dataset). This implies the need for seemingly realistic appearing obfuscation. For example, Howe, Nissenbaum and Toubiana (2008) [17] developed a browser extension, TrackMeNot, that obfuscated user identities from search engines by adding randomly selected search queries from other users. This method adds realistic "fake" queries to a user's query history. If the "fake" queries were not realistic at all, an attacker could easily remove all "fake" queries from the history and thereby reverse the obfuscation.

In this study, we present a successful obfuscation algorithms that preserve the quality of the data while being immune against gender-inference attacks on user-item matrices. Our algorithms extend the already existig, state-of-the-art obfuscation algorithm BlurMe, proposed by Weinsberg et al. (2012)[50]. The user-item matrix represents all interactions (i.e. ratings) between users and items. This study investigates the potential of the gender inference attacks on the user-item matrix, which contains ratings of users for different movies. We evaluate different obfuscation approaches: (1) the one presented by Weinsberg et al. (2012)[50], called BlurMe, (2) our novel approach, called BlurM(or)e and (3) a variation of BlurM(or)e, called BlurMeBetter. BlurM(or)e is an extension of BlurMe and proves to be better at securing the privacy of users. However, BlurMe and BlurM(or)e seem to appear quite artificial in the eyes of a machine learning classifier. That means that an attacker can still identify obfuscated datasets and can take steps toward reversing the obfuscation. To prevent that, we propose a variation of BlurM(or)e, called BlurMeBetter, which obfuscates the gender of a user successfully, while maintaining a realistic appearance and data quality.

This study addresses three main parts. Each part aims to look at a different research

question: In the first part, we investigate different attributes in the user-item matrix of the MovieLens dataset. We aim to find the attributes with the highest contribution toward successful gender inference attacks. The knowledge we gather in this section will be used to develop the BlurMe extensions, BlurM(or)e and BlurMeBetter. Thus, the research question of part 1 is:

> Which factors or attributes make the data vulnerable to gender inference attacks?

The second part shows a major flaw in the obfuscation algorithm of Weinsberg et al. (2012)[50], which is currently considered the state-of-the-art. We show that it is possible to identify data that was obfuscated with the BlurMe approach via simple visualization techniques. As a result, it is still possible to extract the original gender information of users, leaving the private information of users in the presumably obfuscated dataset unprotected. We propose an extension to BlurMe, called BlurM(or)e and investigate if BlurM(or)e is equally capable of gender obfuscation while maintaining the quality of the dataset, and being more difficult to identify as obfuscated. The second research question would thus be:

> How can BlurM(or)e be capable of gender obfuscation, while maintaining the quality of the data, and producing a dataset that is more difficult to identify as obfuscated?

Finally, the third part takes a more complex approach to determine if a dataset is obfuscated or not. We show that it is possible to train a classifier that can distinguish between original user profiles and obfuscated user profiles. As a result, we extend the BlurM(or)e approach and present an adversarial approach to obfuscate the "fakeness" in the gender-obfuscated dataset and another approach that obfuscates gender in a smarter way. We again compare the new approaches with the BlurMe and BlurM(or)e approach to validate the gender obfuscation and the more realistic appearance. The third research question is thus:

> How can we obfuscate the gender in a dataset while maintaining the dataset quality and a realistic appearance?

In more general terms, this study contributes to the research question "How can we protect users' demographic information from gender inference attacks while maintaining the accuracy of recommender systems?". It offers novel possibilities that make data sharing possible without privacy concerns.

The contribution of this study is a detailed analysis of different data attributes that contribute to successful gender inference attacks. It is shown that only a fraction of the user-item matrix is necessary for gender inference (section 4). Certain movies and genres have a higher contribution to the gender inference than others. In addition, this study presents an attack to infer the gender of users from a dataset obfuscated with the BlurMe algorithm. It is shown that the gender of a user is unprotected, even after applying the BlurMe algorithm (section 5.2). Moreover, we propose BlurM(or)e, which is more robust against the proposed attack. It obfuscates the MovieLens dataset successfully, while retaining an acceptable data quality. Finally, and most remarkably, we contribute an even better algorithm, BlurMeBetter, that is capable of gender obfuscation, while maintaining a realistic appearance of the dataset.

The paper is organized as follows. In section 2 we give an informative overview of the necessary background information and related work, that is connected to this study. The next section, section 3, contains the replication of the gender inference models and obfuscation techniques used in Weinsberg et al.(2012)[50]. After that, the next three

sections, section 4, 5 and 6, contain the three major parts mentioned above. We finish in section 5.7 with a discussion and conclusion. All coding was done in Python 3.7 and can be found on GitHub at `https://github.com/STrucks/BlurMore`.

2 Background & Related Work

2.1 Recommender Systems

Increasingly employed by online platforms, recommender systems involve algorithms that provide personalized product or service recommendations. Technically speaking, such a system is an information-filtering framework that aims to predict the user's affinity towards a certain item i.e. product. The goal of a recommendation system is to facilitate the choice process for users, guaranteeing that only a small amount of the most relevant items are displayed (Resnick and Varian, 1997 [39]). Netflix, for instance, displays movie genres that are most likely of personal interest to the respective viewer, whereas dating websites provide partner suggestions that match the user's interests. Recent exploratory studies confirm the success of recommender systems. MacKenzie, Mayer and Noble (2013) show that 35% of what consumers purchase on Amazon and 75% of what they watch on Netflix are attributable to personalized recommendations [30]. Likewise, Hassler (2017) observes that 11.5% of revenues from online shopping sessions on e-commerce platforms can be attributed to personalized product recommendations [16].

A recommender system contains two primary entities: the recommender, and the user that interacts with the recommender. The recommender receives data from the user, such as ratings for an item, stores and processes it to return personalized recommendations. Aside from the rating data, the recommender also stores meta data, such as the time stamp of the interaction. These metadata contribute to better recommendation quality. To process the user interaction data, the recommender system adopts a certain recommendation strategy like "content-based filtering" (CB) or "collaborative filtering"(CFB). The recommendation strategies are explained in detail in the following section. Figure 1 visually summarizes the system model.

Figure 1: The system model of a personalized recommendation service. The recommender might use the collaborative filtering based (CFB) or content based (CB) strategy. Picture taken from Wang, Zheng, Jiang, and Ren, 2018 [48].

2.2 Recommendation Strategies

CB approaches use the available user information - like gender, age, profession - to find items that could be of high interest for the user, and subsequently recommend these (Pazzani and Billsus, 2007 [36]). In this case, the user demographics must be explicitly given to the recommender system. To exemplify, in the case of movie recommendations, users may state their genre preferences on their profile. For users that like horror movies, these will appear in their list of movie suggestions.

The collaborative filtering approach operates in a different way. Systems that employ the CFB approach exploit similarities in rating behavior among similar users, i.e. those who provided similar information, to predict ratings for unseen items. The idea is that users who behave similar in the past, will behave similar in the future. One well-known

example is the strip of items that is usually placed underneath the product description on Amazon with the headline "Customers who bought this item also bought". A key advantage of the CFB approach is that the recommender system does not need any high-level "understanding" of the products themselves. Recall that the CB approach needs a description of every item, otherwise it cannot be recommended. CFB however relies solely on the interaction data between the user and the items. Of course there are also some disadvantages in the CFB approach. If a new user enters the system, there is not enough data to produce accurate recommendations (this is commonly referred to as the "cold start" problem). Also, big companies like Amazon or Netflix offer millions of products to millions of users. They thus need an enormous amount of computational power to process all the data. This study does not contribute to finding solutions for these problems, but we refer to Schein, Popescul, Ungar and Pennock (2002) [43], Bobadilla, Ortega, Hernando and Bernal (2012) [6], Gouvert, Oberlin and Févotte (2018) [15], or Nilashi, Ibrahim and Bagherifard (2018) [34] for further interest.

Collaborative filtering can be divided into three subgroups. There are memory-based, model-based and hybrid approaches. Memory-based approaches use user ratings to compute the similarity between users or items. Typical examples of memory based approaches are Nearest-Neighbor (see Breese, Heckerman and Kadie, 1998 [7]), deep learning (see Wang, Wang and Yeung, 2015 [49]) and Matrix Factorization (MF) [21]. The experiments in section 5.5 and 6.4 will use MF as recommendation strategy. The key idea behind MF is to represent the users and items in a lower dimensional, latent space (Koren, Bell and Volinsky, 2009 [21]) and became popular after showing good performance during the Netflix Prize challenge in 2006. There are different variations of MF, such as SVD++ (see Cao et al.,2015 [9]; Jia, Zhang, Lu and Wang, 2014 [20]) or Probabilistic Matrix Factorization (see Mnih and Salakhutdinov, 2008 [32]). In our experiments, we will use the basic MF approach that uses singular value decomposition (SVD), as presented in Funk's blogpost during the Netflix Prize challenge [1]. The algorithm decomposes the preference matrix (or user-item matrix), so that

$$P_{m\times n} = U_{m\times m}\Sigma_{m\times n}V_{n\times n}$$

For 4 users and 5 items, the decomposition would take the following form:

$$P_{4\times 5} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{11} & u_{12} & u_{13} & u_{14} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix} \times \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & 0 \\ 0 & 0 & 0 & \sigma_4 & 0 \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \\ v_{31} & v_{32} & v_{33} & v_{34} & v_{35} \\ v_{41} & v_{42} & v_{43} & v_{44} & v_{45} \\ v_{51} & v_{52} & v_{53} & v_{54} & v_{55} \end{bmatrix}$$

with $\sigma_1 > \sigma_2 > \sigma_3 > \sigma_4$. Thus, the preference of user 1 for item 1 can be calculated with

$$p_{11} = \sigma_1 u_{11} v_{11} + \sigma_2 u_{12} v_{21} + \sigma_3 u_{14} v_{31} + \sigma_4 u_{14} v_{41}$$

With this preference score, the recommender system can decide if the item should be recommended or not.

[1] `https://sifter.org/~simon/journal/20061211.html`

2.3 Evaluation of Recommendation Systems

Naturally, we want to know how well the recommendation system predicts the rating of a user for a certain item. In this study, the performance of a recommendation system is evaluated using the Root-Mean-Squared-Error (RMSE) between the predicted ratings and the actual ratings. We choose RMSE because it is perhaps the most popular metric to evaluate the performance of a recommender system (Shani and Gunawardana, 2011 [44]). The system is trained on a training set of user-item pairs and subsequently tested on a testing set (T) of user-item pairs. The true ratings of the user-item pairs in T are known. The RMSE between the predicted ratings y and the true ratings t is calculated as:

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (y_{u,i} - t_{u,i})^2}$$

where $|T|$ denotes the size of the training set. Thus, a small RMSE means that the recommendation systems' predictions are accurate and thereby, the system performs well. In this study, we will use RMSE as a metric of data quality. If the RMSE increases after data obfuscation, one can conclude that the data quality decreases and becomes hence less usable.

2.4 Threat Model

Throughout this study, we will present different threat models of an attacker in order to make the hypothetical attack strategy of an attacker clear. The threat model also defines which information leak is considered to be a privacy breach. Following the adversarial threat model scheme presented by Salter, Saydjari, Schneier and Wallner (1998) [42], the threat model addresses three major characteristics of an attack:

Firstly, it specifies the **resources and objectives (Goal)** of the attacker, describing what the attacker is capable to do that poses a threat and what the goal of the attacker is, e.g. gender inference attacks to infer the gender of users without their consent. Secondly, the threat model defines the **vulnerability**, which makes the attack possible, e.g. possession of a third party dataset. Lastly, it specifies the **countermeasure**, which would be the solution to prevent the attack.

2.5 Privacy in Recommender Systems

Although recommender systems are of great benefit, there are also some privacy issues needing attention. In regard to the Netflix Prize challenge, we have seen that it is possible to identify users from anonymized user-item interaction data [33]. This is of course a concerning privacy breach and suggests that better methods for securing privacy in recommender systems need to be developed. This section presents different approaches that contribute to more effective securing of privacy.

First of all, we need to specify what privacy exactly is in this context. Many researchers have different opinions about the definition of privacy, but we will take the definition of Craig and Ludloff (2011) [10] from their book *Privacy and Big Data*. As the authors state, there are three different aspects of privacy:

- **Physical privacy:** No one is allowed to intrude your physical space (e.g. your property).
- **Information privacy:** No one is allowed to collect, store, share or process your personal information without your consent.

- **Organization privacy:** No third party is allowed to access your personal data stored at a trusted organization.

This study focuses on *Organization privacy*. As mentioned before, the collection and storing of user-interaction data contains personal information about the user. Sharing that data without the user's consent would thus conflict with his or her organization privacy.

As Tang (2017)[45] points out, protecting the users' private information should be of high priority for organizations or companies for various reasons, such as:

- **Protecting Non-Production Data:** Since most companies use copies of production data (like the rating data in the MovieLens dataset) for non-production purposes (e.g. Application training and testing; personnel training; business analytics modeling), a dataset will be shared with an considerable number of people. If the dataset is not masked, all these people have access to the private information hidden in the dataset. The risk of the dataset falling into the wrong hands is enormous. If the dataset is not protected, it might be accessed by contractors or offshore workers, and be shared again via the cloud or removable media. Thus, sharing unprotected data makes it possible for an attacker to abuse the dataset and to breach privacy.

- **Protecting Against Insider Threats:** Not only workers from outside should use the protected data, but also trusted employees that do not necessarily need the unprotected data. A study in 2016 from the Risk Based Security Incorporation reveals that 18.3% of data breach incidents were the result of insider activity[1]. Clearly, masking sensitive data would prevent the private information to spread.

- **Complying with GDPR:** Since May 2018, the General Data Protection Regulation (GDPR) has taken effect. This regulation intends to strengthen and unify personal data protection. If the conditions of the GDPR are not met, the sanctions can include a fine up to 20 million Euros. The article 32 of the GDPR introduces the key concept of pseudonymization, that means that companies must anonymize their data by replacing personal identifiers or sensitive information (e.g. names) by realistic, but fictitious, data.

Past research addressed this problem and suggests to use data encryption, differential privacy or data obfuscation techniques. Data encryption relies on encrypting the user interactions so that the recommender system has to work with encrypted data. This ensures high security for users, but working with encrypted data leads to high computation overhead, especially for large-scale data. To avoid this overhead, researchers came up with differential privacy techniques. The key concept of differential privacy is the goal of maximizing the response-accuracy for a certain query, while protecting the information of the data entries that were used to answer the query. This study addresses data obfuscation, but we refer to Erkin, Veugen, Toft and Lagendijk (2012) [14]; Badsha and Yi and Khalil (2006) [3]; Badsha, Yi, Khalil and Bertino (2017) [4] for data encryption and to Dwork (2011) [13] for differential privacy for further interest.

Data obfuscation (aka data masking) describes the process of hiding sensitive data, with modified or even fictional data (Durlach et al., 2003 [12]). Naturally, successful data obfuscation comes at the cost of data utility. Data obfuscation can be done in several ways: Reddy and Knight (2016) [38], for example, use lexical substitution as obfuscation mechanism. In their study, they try to infer the gender of an author on the basis of Twitter and Yelp data. Twitter and Yelp are platforms for users to write short texts about anything or reviews about restaurants respectively. Reddy and Knight found that there are some words that are indicative for the user's gender, like *bro* or

lol for males, and *love* or *boyfriend* for females. Hence, they obfuscated the data by substituting some male words with female words and vice versa. As a result, the authors were able to successfully fool the classifier: If they modify a male text, the classifier will misclassify it as a female text. However, the semantic meaning of the text is sometimes lost, which is a major drawback.

In the context of movie recommendation, the movie ratings may contain hidden and possibly sensitive personal information about the user. As Narayanan and Shmatikov (2008)[33] show, one can find out e.g. the users' identity or political orientation from high-dimensional micro-data such as movie ratings data. This is obviously a privacy breach and makes online data sharing unfeasible from a legal and ethical perspective. Yet, data sharing is important to create online competitions that aim to improve state-of-the-art recommendation systems (see for instance Bell and Koren [5]) or for scientists that must have the same dataset to compare their experimental results (Tenopir, 2011 [46]).

2.6 Gender in a User-Item Matrix

Obfuscating the User-Item matrix: In order to protect user demographics information in the user-item matrix, researchers have suggested data obfuscation. The goal is to protect the privacy of users, while maintaining the utility of the data. In Weinsberg et al. (2012)[50], further referred to as "BlurMe", the authors found that it is possible to infer the gender of users from their rating histories via basic machine learning classifiers. They then proposed an algorithm, BlurMe, that successfully obfuscates the gender of a user and thereby blocking gender inference. BlurMe basically adds ratings to every user profile that are typical for the opposite gender, and is currently state-of-the-art. The best performing BlurMe obfuscation strategy, the *greedy* strategy, decreases the accuracy of a logistic regression inference model from 80.2% on the original data to 2.5% on the obfuscated data (adding 10% extra ratings). The other proposed strategies have a smaller impact on the classification accuracy. Details and more explanations about the gender inference and obfuscation process can be found in section 3.2.

Inference on the User-Item matrix: The goal of BlurMe obfuscation is to protect against gender-inference in the user-item matrix. The most recent work on inference on the user-item matrix is, to our knowledge, that of Liu, Qu, Chen and Mahmud (2019) [28], who developed a deep retentive learning framework that beats the conventional, standard machine learning approaches in the task of inferring user demographic information. For gender inference, Liu et al. [28] achieve a classification accuracy of 82%. However, this is only 2% better than the standard logistic regression model used in BlurMe [50]. We adopt the model from BlurMe in this study since it is sufficiently close to the state-of-the-art for our purposes.

2.7 Realistic fake data

Previous studies address privacy-preserving techniques, which modify existing data or even add fictional data to hide personal information. Krumm (2007) [22] shows that it is possible to infer the identity and home of somebody, solely based on geographic GPS data. Among 172 volunteer subjects, Krumm was able to infer their home location with an error of about 60 meters. He then used a white pages look-up to find out the names of the user. Although Krumm found only 5% of the correct names, it is most likely that this inference will be more successful once GPS signals will become more precise. To counter this inference attack, Krumm presents different obfuscation techniques that rely on perturbation of the data. However, none of the proposed techniques seem to be

robust against the inference attack. Only extreme perturbations preserve privacy, but Krumm points out that in these cases the data becomes unusable. Future work could thus investigate methods to obfuscate the home location with authentic fake data, to keep the usability of the dataset intact.

In August 2006, AOL released detailed search histories of their users. Although the search queries seemed to be anonymous, some of them contained enough personal information so that it was possible to infer the identity of several users. In response to that incident, Howe, Nissenbaum and Toubiana (2008) [17] developed a browser extension called TrackMeNot, that obfuscates the identity of a user by injecting fake queries to the search engine. The goal is to hide the user's search trails in the mass of fake queries. Of course, the fake queries must be realistic, otherwise it would be easy to remove the fake queries. Howe et al. (2008) archive that by using random search queries that other users queried before.

3 Experimental Setup

3.1 Data

MovieLens 1m: During our experiments, we will use the MovieLens 1 Million dataset, publicly available at `https://grouplens.org/datasets/movielens/`. This dataset is published by GroupLens, a research team from the Department of Computer Science and Engineering at the University of Minnesota. It is commonly used as a benchmark for research with recommender systems. Weinsberg et al. (2012)[50], whose study we are going to replicate, used the same dataset. The MovieLens dataset comprises 6 thousand users that produce 1 million ratings on 3.7 thousand movies. In the MovieLens dataset, each user has a reference to its gender, occupation, age and postcode. Since Weinsberg et al. (2012)[50] focus exclusively on the binary gender of the users, we will do the same. It is important to note that the distribution in the dataset is unbalanced: there are 4331 males that produced 750K ratings and 1709 females that produced 250K ratings. The ratings range from 1 to 5 and every user has at least rated 20 movies.

Datasets for generalizability: To check if the effects of the experiments also generalize to other datasets, we will repeat the experiments on two other datasets. The first one being the Flixster dataset, which is very similar to the MovieLens dataset. The main difference is that the Flixster data contains more users, more items and more interactions. It comprises about 1 million users that produce 16 million ratings on 66 thousand movies.

Aside the Flixster data, we will also use the LibimSeTi dataset, which is from a completely different domain (Brozovsky and Petricek, 2007 [8]). That dataset was released by the Charles University in Prague and originates from the Czech dating website `LibimSeTi.cz`. It contains about 17 million ratings of 169K profiles made by 135K users. More details about the dataset statistics can be found in table 1.

Due to the size of these datasets (see table 1) and the given computational constraints, it is only possible to take a randomly selected subset of 400 users from both, the Flixster and the LibimSeTi dataset. Table 1 displays some general statistics of the three datasets.

dataset	#Users	% Males	#Items	#Ratings	Av.rating	Density	Variance
MovieLens 1m	6,040	0.72	3,706	1,000,209	3.58	4.47%	1.25
Flixster	1,002,796	0.53	66,730	16,392,155	3.62	11.38%	1.2
LibimSeTi	135,359	0.55	168,791	17,359,346	5.94	7.60%	9.68

Table 1: Statistics of all datasets that will be used in our experiments.

3.2 Re-implementing BlurMe

Since we did not have the code of the original BlurMe, we re-implemented it in order to carry out the comparison in this work. Because the paper was not specific about the settings of all parameters, it is not possible to create an exact replication. For completeness, we discuss our re-implementation here, so that authors building on our work have the complete details. All of our coding was done in Python 3.7 and we used different libraries, such as the sklearn-package (Pedregosa, 2011[37]).

BlurMe's Gender Inference: This paragraph describes our re-implementation of the gender inference models.

We create the user-item matrix by associating every user with a vector of ratings: x_i with i being the index of the user and $x_{i,j}$ being the rating of user i for movie j. If the movie is not rated, we set $x_{i,j} = 0$. This results in a $U * I$ matrix, where U is the

number of users and I is the number of items. Every user vector is associated with a gender, that will serve as target label for the classifier.

Following the experiments of Weinsberg et al. in [50], all classifiers are trained and tested on this user-item matrix with 10-fold cross-validation. We do not have information about their splits, so we use our own splits. The ROC area under the curve as well as precision and recall are reported as performance measures. A comparison of the results can be seen in table 2. The SVM uses a linear kernel and a C value of 1. For the Bernoulli classifier, the user-item matrix is transformed, so that every rating $x_{i,j}$ that is greater than 0, is set to 1. This means that the Bernoulli Bayes classifier ignores the value of the rating and only uses information about whether a user i rated the movie j or not. All remaining parameters for the other classifiers are set to the default values. There is about a 4% difference between the scores reported in the original BlurMe paper, and those we measured with our reproduction.

	BlurMe results		Replication results	
Classifier	AUC	P/R	AUC	P/R
Bernoulli	0.81	0.79/0.76	0.77±0-02	0.88±0-02/0.48±0.05
Multinomial	0.84	0.80/0.76	0.81±0.02	0.89±0.02/0.77±0.05
SVM	0.86	0.78/0.77	0.79±0.02	0.83±02/0.82±0.02
Logistic	0.85	0.80/0.80	0.81±0.02	0.84±0.02/0.83±0.02

Table 2: Comparison of the replication and original results from Weinsberg's et al. (2012)[50] of the gender inference model on the MovieLens dataset. The performances are measured in ROC AUC (AUC), precision (P) and recall (R).

BlurMe's Obfuscation: This section describes our re-implementation of the obfuscation approach of BlurMe [50]. Recall that the basic idea of BlurMe is to add fictional ratings to every user that are atypical for the respective gender. BlurMe creates two lists, L_f and L_m, of atypical movies for each gender by training and cross-validating a logistic regression model on the training set. The movies in L_f and L_m are ranked according to their average rank across the folds. The rank of a movie within a fold is determined by its coefficient that is learned by the logistic regression model. The lists L_f and L_m also include the average coefficient over all folds for each movie that serves as correlation metric between the movie and the user's gender.

After these lists are created, they take every user profile and add **k** fictive ratings to the profile for movies from the opposite gender list. The parameter **k** limits the number of extra ratings and is set to 1%, 5% or 10% in their experiments. A male user with 100 ratings in the original dataset would be obfuscated by adding 5 (for **k**=5%) fictive ratings from the female list.

There are some design choices left: Which movies should be selected from the lists and what should the fictive rating be? Weinsberg et al. (2012)[50] proposed three different selection strategies for the first problem: the *Random Strategy*, the *Sampled Strategy* and the *Greedy Strategy*. The *Random Strategy* chooses **k** movies uniformly at random from the list, the *Sampled Strategy* chooses **k** movies randomly, but in line with the score distribution of the movies. Thus, a movie that has a high coefficient is more likely to be added. Finally, the *Greedy Strategy* chooses the movie with the highest score. The authors do not mention the length of the lists, thus we chose to include all movies with a positive coefficient in the L_f list, and all movies with a negative coefficient in the L_m list.

For the fictive rating of a user A for a movie B, BlurMe suggests using either the global average rating for movie B or the predicted rating for user A for movie B. Since Weinsberg et al. (2012) [50] reports that there is almost no difference between these

approaches, we chose to set the extra ratings for a movie according to its respective overall average rating. This average is rounded, because only integer ratings are valid. Figure 2 visualizes the BlurMe algorithm for one user.

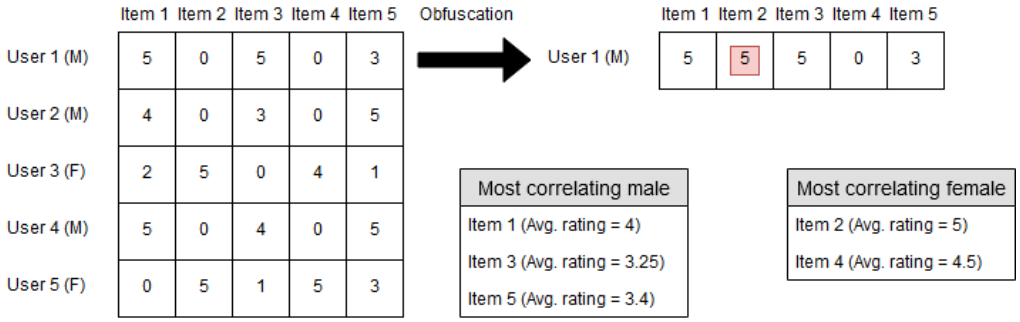

Figure 2: The BlurMe algorithm visualized for one user. User 1 is obfuscated by adding the rating "5" to item 2, because 5 is the average item rating for item 2, which is the first item from the table "most correlating female". This example would use $k = 33\%$ extra ratings with the greedy strategy.

The authors of BlurMe take the following attack protocol into account: A gender inference model is trained on real, non-obfuscated data and tested on our obfuscated data. Therefore, the gender inference model is trained on unaltered data and tested on obfuscated data. They use 10-fold cross-validation and report the average classification accuracy of the model.

We report results achieved by our BlurMe reproduction in table 3. The reproduction is generally congruent with the original. The difference is negligible, we can see that the classification accuracy decreases if the obfuscation increases. Further exploration revealed that normalization, in terms of scaling all ratings from values in $[0, 5]$ to values in $[0, 1]$, can have a large impact on scores. We do not focus on normalization further here, but point out its impact because it suggests that there are parameters that could have been adjusted that are not explicitly recorded in the paper.

		Extra ratings			
Algorithm	Strategy	0%	1%	5%	10%
BlurMe	Random	0.80	0.78	0.72	0.61
	Sampled	0.80	0.75	0.59	0.36
	Greedy	0.80	0.58	0.17	0.03
Reproduction	Random	0.76±0.02	0.74±0.02	0.69±0.03	0.62±0.04
	Sampled	0.76±0.02	0.71±0.02	0.58±0.04	0.33±0.04
	Greedy	0.76±0.02	0.54±0.03	0.15±0.03	0.02±0.01
Reproduction Normalized	Random	0.81±0.02	0.80±0.02	0.78±0.01	0.76±0.02
	Sampled	0.81±0.02	0.80±0.02	0.78±0.02	0.75±0.01
	Greedy	0.81±0.02	0.78±0.01	0.74±0.02	0.70±0.02

Table 3: Performance of BlurMe's and the reproduction's obfuscation algorithm on the MovieLens data measured in classification accuracy with standard deviation across the 10-folds. The original BlurMe paper does not provide the standard deviations.

Note that table 2 uses ROC AUC as performance metric and table 3 uses classification accuracy. This choice was made by BlurMe and for the sake of comparing the models, we did the same.

4 Part 1: Relevant Attributes for Gender Inference

This section investigates detailed properties of the dataset and tries to answer the first research question "Which factors or attributes of the data contribute to successful gender inference attacks?". In this section, different factors and attributes of the data are evaluated on their importance for the gender classifier. Knowing which factors contribute toward correct gender prediction is essential for designing a powerful obfuscation mechanism. Weinsberg et al. (2012)[50] have only a very limited contribution to this topic, i.e., they present the top-10 movies from the Flixster dataset that correlate most with either of the genders. This study however takes an in-depth exploratory approach by presenting a detailed overview about several attributes of the data and their respective importance to the gender inference model.

4.1 Data Saturation

Firstly, it is interesting to find out how many ratings a gender classification algorithm needs to have for a user to infer his or her gender. To answer this, the gender classifier from above is trained again, but this time, the number of ratings per user is restricted. In table 4, the performance of the classifier is reported for varying maximum ratings per user. In the first column ($n \leq 1$), for example, the user-item matrix contains only one rating per user. This rating is selected randomly and all other ratings are dropped/ignored. Since this procedure involves randomness, table 4 contains the average AUC score with standard deviation for 10 trials.

max ratings	$n \leq 1$	$n \leq 5$	$n \leq 20$	$n \leq 100$	$n \leq 200$	all
AUC	0.54±0.04	0.61±0.01	0.69±0.02	0.78±0.02	0.80±0.03	0.81±0.02
Precision	0.74±0.01	0.77±0.01	0.81±0.01	0.84±0.02	0.85±0.02	0.84±0.02
Recall	0.89±0.01	0.80±0-01	0.80±0.02	0.84±0.02	0.84±0.03	0.84±0.02

Table 4: Logistic regression with limited number of ratings per user using the user-item matrix of the MovieLens dataset.

The results from table 4 show that the classification algorithm - at least in the case of logistic regression - does not need more than 200 random ratings per user to infer the gender of a user. After 200 movies, every additional movie rating has diminishing returns. One can also see, that the algorithm can classify the gender by only using 20 ratings per user (AUC = 0.69). That means that after only rating 20 random movies, one can infer your gender with (medium) success. Although this already shows that only a fraction of the dataset is necessary to infer the user's gender, we will show that one can use even less data to do the same task with more success.

4.2 Binary User-Item Matrix

Next, we take a closer look at the importance of the ratings. Naturally, one might assume that the rating reveals gender information, because male users might rate a certain item with a "5", while female users would rated the same movie with a "1". To test if that assumption is met, we will transform the user-item matrix into a binary matrix. In essence, we will replace every rating with a "1", thus the new matrix only contains information about whether a user watched a certain movie or not. The matrix does not contain any information about whether a user liked a certain movie or not. We train and test the gender inference model with the new binary user-item matrix and achieve an AUC of 0.81. Surprisingly, the exact ratings of movies seem to be irrelevant. In fact, the AUC for the binary user-item (U-I) matrix is the same as the original U-I matrix and

thus, gender inference is possible . That means, that the gender inference model only needs information whether a movie has been watched or not. This information must already hint toward the gender of a user. The effect provides evidence that obfuscation should alter the zeros in the user-item matrix.

4.3 Indicative movies

Weinsberg et al. (2012)[50] already found out that there are some movies that correlate with the user's gender. The computed coefficients of the logistic regression classifier reveal the correlation between movies and gender. They present the top 10 movies that correlated the most with either gender. This suggests that there are some movies that have a higher contribution to the gender classifier than other movies. This makes sense, because one can imagine that a movie like "Terminator" is more frequently rated by men and less frequently rated by women. Thus, if we know that a user rated the movie "Terminator" with only one star or (or even 0 stars - meaning that the user did not watch the movie at all), we can assume that the user is most likely female.

To test that, we will run the gender classifier from the previous experiments again, but this time with a modified user-item matrix. The idea is to only keep the movies in the user-item matrix that are indicative for the user's gender and remove all movies that are not indicative from the user-item matrix. If the performance of the classifier stays the same, one can assume that the remaining movies in the user-item matrix provide all information necessary about the user's gender. The movies that were removed are thus not indicative of gender. If, however, the performance drops substantially, one can assume that the movie selection process removed all important movies for gender classification. That means, if the AUC drops, we remove movies that are actually indicative. The goal is to find the subsets of movies that do and do not contribute to the gender classifier. Having these two subsets, we would get an impression about the percentage of relevant movies in the dataset. Knowing this percentage might help during the obfuscation step, later in section 5.

Intuitively, features that show a different average rating for males than for females, can be seen as gender-indicative. We can compute the difference of the average rating per movie for both gender and test that difference for significance. This is done with a two-sided t-test against the null hypothesis of no difference in average rating data across genders. Since we run this test for all 3.7K movies, we have to correct for the alpha error, by using the Bonferroni correction (Weisstein, 2004 [51]). The resulting p-values must thus be less than $0.05/\textit{number of tests}$, in order to be considered significant. If the test for one movie is significant, that movie can be considered indicative.

After running the tests, we create a new user-item matrix by selecting only the movies with a significant p-value. As a result, we remove about 98% of the data, so that only 76 movies are kept in the U-I matrix. The new U-I matrix is compared with the original U-I matrix by running the gender inference model on them separately. Table 5 displays the performance of the gender inference model using the regular user-item matrix and the user-item matrix with significant movie selection.

Data	AUC	Precision	Recall
regular U-I matrix	0.81±0.02	0.84±0.02	0.84±0.02
sign. movies	0.80±0.02	0.85±0.02	0.84±0.02

Table 5: Comparison of results obtained in our experiments with regular user-item matrix and the user-item matrix with only significant movies using the logistic regression model on the MovieLens data.

One can see that the AUC score drops from 0.81 for the regular user-item matrix to 0.80 for the user item matrix with feature selection. This decrease is not substantial and provides evidence that the selected movies indeed contain gender information. This means that only a small fraction of the data is already enough to classify the gender of a user reliably (AUC of 0.80). Also, one can assume that these 76 movies are highly indicative for the user's gender and thus need obfuscation.

However, they do not contain all the gender information in the dataset. If the complementary dataset is used (the whole user-item matrix, except the 76 movies), the classifier still achieves an AUC of 0.79. Here, the classifier uses 3211 of the 3706 movies (about 87%). There are 419 movies that are not present in either user-item matrices, because they cannot be tested with a t-test due to too few ratings of one gender. This indicates that obfuscating only the 76 movies would not be enough, since the other movies still contain more than enough information to infer the user's gender.

Thus, both of the two subsets contain gender information. We could change the threshold of the p-value until one subset contains all movies that are gender indicative and the other subset contains all movies that are not gender indicative. However, we avoid changing the conventional threshold of 0.05 for a p-value and present a different approach in order to distinguish between indicative and non-indicative movies.

Instead of testing every movie for significance, we can also measure the importance of every movie with the help of a Random Forest classifier. A Random Forest classifier is an ensemble method that uses multiple decision trees for a classification task. It assigns every feature a score, which will be used as a measure for the feature selection in our experiments. This score is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability is calculated by dividing the number of samples that reach the node by the total number of samples. The higher the score, the more important the feature is (Ronaghan, 2018 [40]).

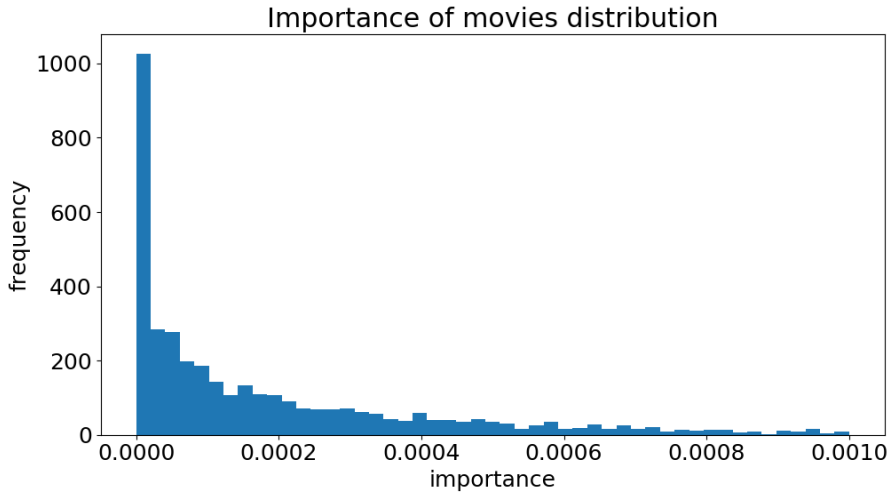

Figure 3: Movie importance distribution from a random forest classifier in the MovieLens dataset. The distribution is cut-off at 0.001 for visualization purposes. The distribution has a classical "long-tail" shape.

Figure 3 depicts the movie importance distribution in the MovieLens dataset. The distribution is cut-off at 0.001 for visualization purposes, but distribution maintains the classical "long-tail" shape also after 0.001. One can see that some movies have a higher

importance score than other. For example, the movie "Sense and Sensibility" (1995) [2] has a score of 0.05 and is hence far more important for a gender classifier than the movie "Cutthroat Island" (1995) [3] with an importance score of < 0.001. This makes sense, because "Sense and Sensibility" is a classical drama or rather romance movie. These genres are favored by females and less preferred by males, as we show later in section 4.4. "Cutthroat Island" on the other hand involves both romance and action elements. These genres are preferred by both, females and males — thus the movie is liked by both genders equally. As a result, such a movie is not indicative for the gender and thus receives a low importance score.

Looking at the importance score distribution, the following question arises: Where should we set the threshold for important movies? Every movie that has an importance score less than the threshold is considered irrelevant and every movie that has a score greater than the threshold is considered important for the gender classifier. Since there is no concrete answer for setting the threshold, we have to find it empirically. Once again, a good threshold would divide the dataset into two groups: One group that consists of only important movies and one group that contains only irrelevant movies. To prove that a threshold is good, the gender classifier should perform very well on the dataset with important movies, while performing extremely bad on the dataset of irrelevant movies. The subset of important movies can later be used for obfuscation. Looking at figure 4 (left), one can clearly see that the AUC of the classifier that uses the irrelevant movies grows with an increasing threshold. Obviously, if the threshold is very high, most movies are considered irrelevant, since most movies have an importance score less than the threshold. The number of irrelevant versus important movies for a certain threshold can be seen in figure 4 on the right. Apparently, there are far more important movies than irrelevant movies, because the gender classifier performs still better than random guessing when using e.g. only the 1000 least important movies out of all 4000 movies (AUC ≈ 0.60).

Beside that, we can also see again that the classifier only needs a small percentage of the data to get an AUC score of about 0.83. This is in line with the previous findings.

[2] `https://www.imdb.com/title/tt0114388/`
[3] `https://www.imdb.com/title/tt0112760/`

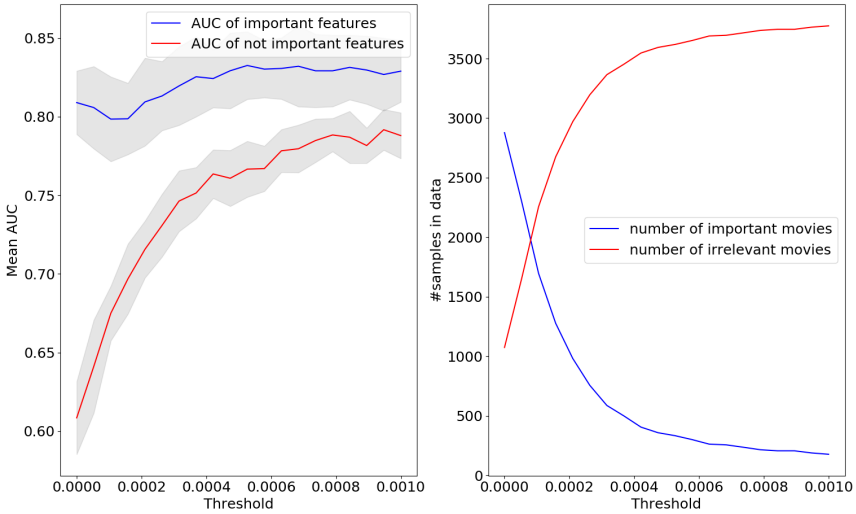

Figure 4: AUC for different thresholds of the dataset (left) and number of users for different thresholds of the dataset (right). The gray area represents the confidence intervals.

BlurMe uses the computed coefficients of a logistic regression model to extract the importance of individual movies. This is a good alternative, because the coefficients represent the correlation of a movie with the gender. Table 6 displays the top-5 most important movie, extracted with the random forest and the top-5 most correlated movies with the male and female gender. Interestingly, the list of most important movies is quite different from the list of most correlating movies. In fact, the top-100 lists only share 4 movies: Persuasion, Little Women, Steel Magnolias and Predator 2. For the obfuscation task, it would be interesting to use the list of most important movies instead of the lists of most correlating movies.

Rank	most important	correlate female	correlate male
1	Sense and Sensibility	Steel Magnolias	Grand Hotel
2	Terminator 2: Judgment Day	Carrie	Simply Irresistible
3	Gone with the Wind	Go West	Hero
4	Dirty Dancing	Kiss the Girls	Godzilla
5	My Fair Lady	Ride with the Devil	Trick

Table 6: Top 5 most important movies and top 5 movies from the lists L_m and L_f for the MovieLens dataset .

4.4 Genre as important attribute

As one can see in section 4.3, not all information in the dataset is needed to infer the gender of a user. Recent research showed that classifiers tend to abstract features hierarchically. For example, artificial neural networks that are used in image classification do not only learn from the individual input features (pixels), but also from higher level/more abstract features (e.g. shapes) (see Lin, Cheng and Yan (2013) [27], Larochelle and Bengio (2008) [24] or Deng and Dong (2014) [11]). In the case of the MovieLens

dataset, there is also a hierarchical structure in the data. Every movie is assigned to one or more genres. Obviously, a genre is an abstraction of a movie, or rather, a romantic movie is a specific element from the collection of romantic movies. Considering that, the following questions rise: Do males prefer different genres than females? Are the genres of rated movies indicative for the user's gender? How much information about a user's preferred genres does the classifier need to classify the user's gender?

Genre Data exploration: To answer the first question, we should be able to find evidence in the data. Figure 5 shows the rating distribution of males (blue bars) and females (orange bars) per gender. The number of ratings for each gender per genre are normalized, thus the amplitude of the bars show a percentage of rated movies per genre. Note that these percentages do not add up to 100, because each movie can have multiple genres. The plot suggests that the genres "Action", "Adventure", "Crime", "Horror", "Sci-Fi", "Thriller", "War" and "Western" are favored by males, while the genres "Children's", "Comedy", "Drama" and "Musical" are favored by females.

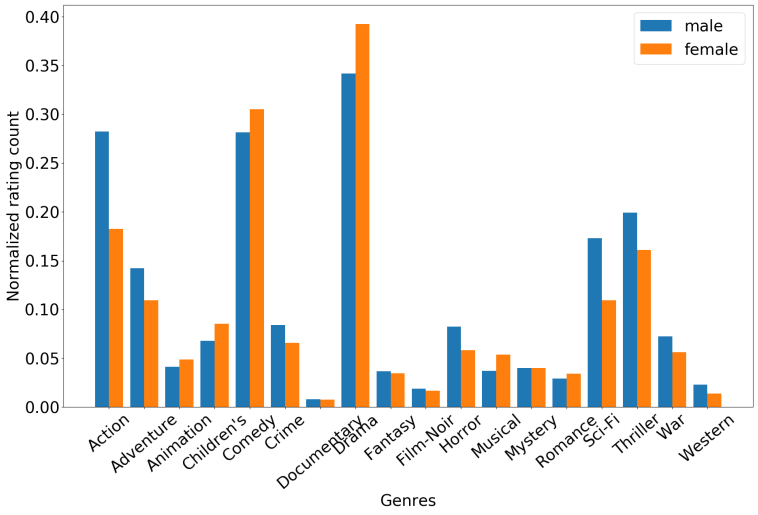

Figure 5: Number of ratings per genre, normalized by the total number of ratings per gender.

Gender Inference in the User-Genre matrix: In order to test if we can train a gender classifier that only knows the genres that a given user has rated, we need to transform the user-item matrix. We define a new matrix, called the user-genre matrix, that contains a characteristic vector x_i for every user i, that consists of the genre distribution of rated movies. Imagine a user that has rated five "Action" movies, one "Adventure" movie, ten "Animation" movies, and so on. That user would have an characteristic vector of $x_i = (5, 1, 10, ...)$. These characteristic vectors make up the user-genre matrix and are used for the classification task. Running the gender classifier from section 3.2, we obtain the following results:

AUC	Precision	Recall
0.80 ± 0.02	0.77 ± 0.005	0.96 ± 0.015

Table 7: Performance of the logistic regression gender classifier using the user-genre matrix of the MovieLens dataset.

Clearly, one can see that the classifier can reliably distinguish between males and females (AUC=0.80). One can conclude that an attacker does not have to know the specific movies you are interested in, it is already enough when he has only the movie genres that you are interested in. On the flip side, could the attacker also have enough information if he knew which genres you are not interested in? We can simply test that by counting for each user which movie he has not rated and then accumulate that for every genre. We would thus get another user-genre matrix, but the numbers inside represent the frequency of movies per genre that a certain user did not rate.

Table 8 shows the results for different numbers of stars. The second row corresponds to the case mentioned above: The user-genre matrix is created with **only** the movie genres that were not rated. In the third row, the user-genre matrix is created analogously, using only the one-star ratings. The next row uses only the two-star ratings and so forth.

Ratings	AUC	Precision	Recall
no rating	0.80 ± 0.02	0.77 ± 0.005	0.96 ± 0.015
1 star	0.60 ± 0.02	0.72 ± 0.00	0.99 ± 0.00
2 stars	0.67 ± 0.03	0.73 ± 0.00	0.98 ± 0.01
3 stars	0.73 ± 0.02	0.75 ± 0.01	0.97 ± 0.01
4 stars	0.78 ± 0.02	0.77 ± 0.01	0.96 ± 0.05
5 stars	0.78 ± 0.02	0.77 ± 0.01	0.96 ± 0.02

Table 8: Performance of the gender classifier using the user-genre matrix of the MovieLens dataset.

The most interesting result present in table 8 is that the classifier that uses **only** the movies with no rating is just as good as the classifier that uses all ratings (both have an AUC of ≈ 0.80). Thus, not rating a certain genre provides just as much information as rating a certain genre. This confirms our previous findings that some genres are favored by men (e.g. Action or Sci-Fi) and some genres are not favored by men (e.g. Drama). If a user did not rate (or rarely rate) any drama movies, the user is most likely male. These findings can be compared to the results from section 4.2.

4.5 Discussion - Relevant Attributes

This section presented different attributes of the user-item matrix of the MovieLens data and evaluated their respective contribution to a logistic regression gender inference model in order to answer the first research question.

We have seen that gender inference can be done using only a fraction of the dataset. If we take only 20 random ratings of a user's profile, we can successfully determine the user's gender (AUC=0.69). It is not necessary to consider more than 200 ratings per user, because after 200 ratings, every additional rating has diminishing returns. This underlines the effects found by Larson, Zito, Loni and Cremonesi (2017) [25], who present the principle of minimal necessary data. The authors found a saturation effect as the size of the training set for a recommender system increases. Following the principle of minimal necessary data, one should remove data that has diminishing returns.

Also, this section showed that the exact rating of a movie is irrelevant - it is more important for the gender inference model to know whether someone watched a movie or not. This is in line with Weinsberg et al. (2012) [50], who observe the same effect. In fact, transforming the user-item matrix into a binary user-item matrix reduces the data to "binary, positive-only data", which is used in "one-class collaborative filtering" (OCCF) problems. For example, "likes" on Facebook belong to this filed of recommendation problems, since a user can either like a certain item or not. There has been various research on OCCF, such as Verstrepen, Bhaduriy, Cule and Goethals (2017) [47] or Pan, et al. (2008) [35]. The success of OCCF gives evidence that binary user-item matrices still contain user preference information, which implicitly also contains users' gender information. Projecting this evidence on the later obfuscation task, we can assume that it is not enough to just change the ratings, but rather remove or add fictional ratings.

Next, this section investigated the relevance of individual movies for the gender inference model. Similar to Reddy and Knight (2016) [38], who found that certain words are gender indicative, there are some movies that, if being rated, imply the gender of a user. Using different relevance measures, we showed that gender information is present in most movies. It is difficult to select only a few movies that contain all the gender information. In fact, more than half of the movies seem to be gender-indicative. For the obfuscation task, we thus have a wide selection of movies to add or remove ratings from. Furthermore, it would be interesting to investigate if certain combinations of movies are gender indicative. For example, there might be two movies that seem not very gender indicative on their own, but if these movies co-occure, they might be gender-indicative.

Aside that, this section also showed that preferred movie genres are gender-indicative. Genres like Action, Adventure or Sci-Fi seem to be favored by males, while Romance, Comedy and Drama are favored by females. In fact, it is shown that an attacker can infer a user's gender solely by his genre preferences (AUC=0.80). Of course, it would be nice if the genre preference could be obfuscated as well.

In the next part, we will present the BlurMe obfuscation method, the state of the art gender obfuscation developed by Weinsberg et al. (2012)[50]. We will point out a flaw in that obfuscation method and develop an extension to the BlurMe algorithm, called BlurM(or)e. BlurM(or)e builds up on some of the knowledge gathered in this part.

5 Part 2: Data Obfuscation - Building a Better BlurMe

Data obfuscation (or Data masking) describes the process of modifying data to hide certain attributes. Here, the goal is to protect the gender of a user in a user-item matrix against gender inference attacks, while maintaining the quality of the dataset. Weinsberg et al. (2012)[50] presented the BlurMe obfuscation algorithm, which adds ratings atypical for the user's gender to profile. Using that algorithm, it is possible to decrease the gender-inference accuracy of a logistic regression model from 80% to 2.5%.

This section investigates the BlurMe algorithm, points out a flaw and presents an extension of BlurMe, called BlurM(or)e. We present a possibility to reverse the BlurMe obfuscation and to thereby breach the privacy of users in the user-item matrix. BlurM(or)e, in contrast to BlurMe, proves to be more resistant against that possibility. Thus, BlurM(or)e seems to secure the user's privacy more effectively. We compare both algorithms in terms of obfuscation performance, maintenance of data quality, and generalizability.

5.1 Threat Model

Using the terms of section 2.4, BlurMe uses the following threat model:

Resources: An attacker encounters a user-item interaction dataset that is not obfuscated. He or she has a pretrained gender inference model, which can be used to infer the gender of the users from the encountered dataset.

Goal: The attacker will use the pretrained gender inference model and reveal the gender of all users from the newly encountered dataset. This information will e.g. further be sold to a third party.

Vulnerability: The attacker can infer the gender of the users' without their knowledge and thereby breach their privacy.

Countermeasure: The encountered dataset will be obfuscated, so that the pretrained model cannot infer the users' gender. The inferred gender information will thus become useless for a third party.

This threat model assumes that both datasets have the same movies. This assumption is reasonable, because the attacker can use only the movies that are present in both datasets. For the sake of comparison, we will use the same threat model later in the obfuscation experiments in section 5.4.

5.2 The Issue with BlurMe

As shown in section 3.2, BlurMe proposes a powerful algorithm that can obfuscate the gender of a user. However, BlurMe has an important flaw: If the rating frequency of the movies is visualized, it is possible to determine that BlurMe has been applied to the dataset, and to identify the movies for which ratings have been added. In figure 6(A), the rating frequency is shown for 20 items from the MovieLens dataset before obfuscation. In figure 6(B), the rating frequency is shown for the same 20 items after obfuscation with BlurMe. BlurMe exhibits sharp spikes of items, in this case, it is item ID 27 (called *Persuasion*), which is marked in red. These spikes indicate that BlurMe has been applied, and point to the movies for which ratings have been added. There are two dangers associated with these spikes. First, if BlurMe is running at an operating point

of 10% extra ratings using the greedy strategy, as mentioned above, then the gender inference accuracy is 2.5%. This means that if the information is known that BlurMe has been applied, it is simple to reverse the decision of the classifier, and gender can be determined with an accuracy of 97.5%. Second, if we do not know the operating point of BlurMe ($< 10\%$ extra ratings will not guarantee us a gender classification accuracy that we can reverse), we still can find the spikes in the rating histogram, and attempt to reverse BlurMe. In order to find a BlurMe spike we would look for movies that are known not to be particularly popular, but still have a lot of ratings in the BlurMe data. This information would have to be gathered from external databases (like the IMDB). In this study, we focus on addressing the first danger, and leave the second to future work.

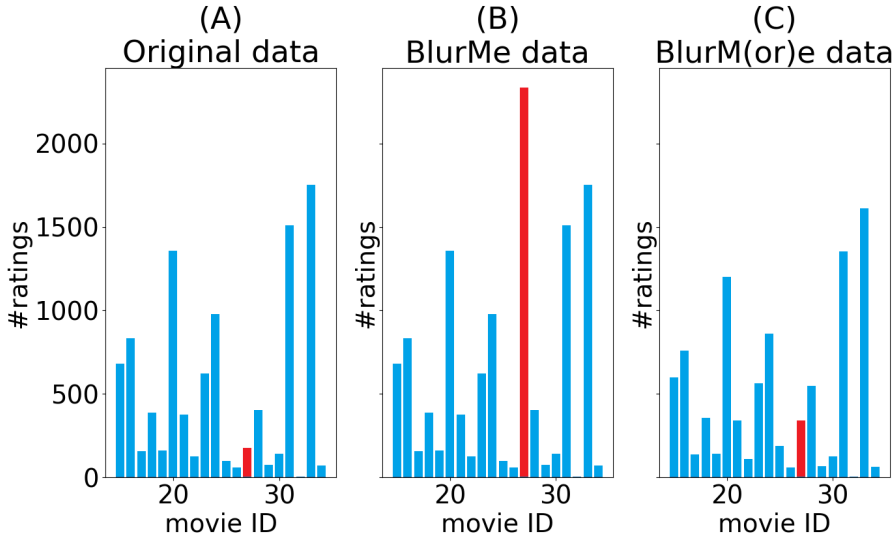

Figure 6: #ratings per movie for the movies 15 to 35. The red bar indicates an example of obvious data obfuscation after BlurMe is applied. The BlurMe data was created with the greedy strategy and with 10% extra ratings. The BlurM(or)e data contains also 10% extra ratings.

5.3 The Definition of BlurM(or)e

BlurM(or)e was inspired by the exploratory analysis that we carried out in the previous part, section 4, which revealed that a large number of movies are indicative of gender. For this reason, it is not necessary to restrict the algorithm to add ratings only to the most correlated movies of the opposite gender (like the greedy strategy of BlurMe does). This means that we can mask the data without heavily relying on a small set of movies indicative of the respective opposite gender. Based on these insights, we designed BlurM(or)e, which works as follows: We create, just like BlurMe, two lists of movies, L_f *and* L_m, that correlate most strongly with females and males respectively. After that, we alter every user profile by adding movies from the opposite gender list with the greedy strategy proposed in BlurMe. However, if a movie has already doubled its initial rating count, it will be removed from the list and will thus not be used for subsequent users for obfuscation. We use $\times 2$, i.e., doubling, in this study because it works well, and leave exploration of other possible values to future work. This increases the variety of movies that receive extra ratings. Also, we keep track of the number of added ratings, so that we can remove the same number later on. After every user has received extra ratings up to a fixed percentage of their original ratings, we remove ratings from users that have

rated a lot of movies (here we choose $\geqslant 200$ movies, because section 4.1 reveals that after 200 ratings per user, every rating has diminishing returns anyway). This removal would be more difficult to diagnose in the user-item matrix, since exact information of the rating rates about users would need to be available. Our method thus adds ratings to a greater variety of movies, because the algorithm does not start at the most correlated movie for every user.

5.4 Comparison of BlurMe and BlurM(or)e

We compare the performance of our new obfuscation mechanism, BlurM(or)e, with the original obfuscation mechanism BlurMe. The performance is measured, in line with the experiments in BlurMe, in classification accuracy of a logistic regression model, trained on unaltered data, and tested on obfuscated data. The performance is cross-validated using 10-fold cross-validation. Table 9 shows that BlurM(or)e performs similar to BlurMe. The gender obfuscation is weaker when applying BlurM(or)e, but one can see that the more obfuscation is applied, the smaller the inference accuracy is. Note that table 9 contains the reproduction of BlurMe that is discussed in detail in section 3.2. We refer a dataset which is obfuscated by BlurMe or BlurM(or)e as the "BlurMe" and "BlurM(or)e" dataset respectively. Of course one could decrease the inference accuracy even more by adding 20% or more extra ratings.

		Extra ratings			
Dataset	Classifier	0%	1%	5%	10%
BlurMe	Logistic Regression	0.76±0.02	0.54±0.03	0.15±0.03	0.02±0.01
BlurM(or)e	Logistic Regression	0.76±0.02	0.64±0.03	0.36±0.07	0.19±0.07
Original	Random Classifier	0.50	0.50	0.50	0.50

Table 9: Gender inference results measured in accuracy on BlurMe (reproduction) and BlurM(or)e. The datasets BlurMe and BlurM(or)e are created using the greedy strategy and the MovieLens dataset.

A big advantage of BlurM(or)e is that an attacker cannot easily see that the dataset is obfuscated. Figure 6 on the previous page shows the number of ratings per movie for 20 different movies in the MovieLens 1m dataset. The red bar corresponds to the number of ratings for the movie with ID 27. After the BlurMe obfuscation is applied, the red bar spans approximately ten times its original size. This makes the attacker suspicious and indicates that the dataset is obfuscated. However, if the BlurM(or)e obfuscation is applied, the red bar only doubles its size, which is less noticeable. Also, BlurM(or)e has closer statistics to the original data. Table 10 shows that BlurM(or)e keeps the number of interactions as well as the density similar to the original MovieLens dataset, while BlurMe produces a more dense dataset with more interactions.

The reduction has a less noticeable effect on the dataset. Since the ratings are removed randomly from users with an extreme number of ratings, the number of ratings per movies distribution does not change dramatically (the bar with ID 20 shrinks $\approx 10\%$ of its original size in the BlurM(or)e dataset).

Also, BlurM(or)e does not alter the statistical properties of the MovieLens data as much as BlurMe does. Table 10 displays some statistical properties of the used datasets. One can clearly see that BlurM(or)e is closer to the original MovieLens dataset in terms of total number of ratings and density than BlurMe. The decrease in number of items for the BlurM(or)e datasets is due to the random removal of ratings.

dataset	#Users	#Items	#Ratings	Av.rating	Density(%)	Variance
MovieLens 1m	6040	3706	1.000.209	3.58	4.47	1.25
BlurMe 1%	6040	3706	1.013.416	3.58	4.53	1.25
BlurMe 5%	6040	3706	1.052.886	3.58	4.70	1.20
BlurMe 10%	6040	3706	1.099.545	3.57	4.91	1.16
BlurM(or)e 1%	6040	3705	1.000.797	3.57	4.47	1.24
BlurM(or)e 5%	6040	3700	1.000.773	3.55	4.48	1.22
BlurM(or)e 10%	6040	3699	1.000.395	3.57	4.48	1.16

Table 10: Statistics of the datasets used in our experiments and analysis. The percentages denote the percent of extra ratings added to the dataset.

5.5 Recommendation performance

Even though the obfuscation is effective, we need to look at the performance of a recommender system to check whether the data utility is maintained. Just like BlurMe, we use the well-known and commonly used collaborative filtering technique, Matrix Factorization (Bell and Koren, 2009 [21]), and notice that the change in RMSE is not substantial. The change has a maximum of 0.0298 for MovieLens with BlurM(or)e and 0.0381 for BlurMe (with greedy strategy and 10% extra ratings). We can see in table11 that the RMSE is decreasing with an increase in obfuscation. Weinsberg et al. (2012)[50] discovered the same effect and explained that this might be due to the density of the obfuscated data. Since BlurM(or)e does not increase the density of the dataset, it must have another reason. While Weinsberg et al. (2012)[50] suggest that the RMSE decreases because the extra ratings contain some meaning, we suggest that adding ratings to users with only very few ratings contributes to better recommendation predictions. An attribute of the MovieLens dataset is that every user has at least 20 rated movies. If we add 10% extra ratings, we increase that number, so that every user has at least rated 22 movies. This increase might reduce the RMSE, since it is difficult for recommender systems to provide accurate recommendations to users with few data (Ahn, 2008 [2]).

	Extra ratings			
Obfuscation	0%	1%	5%	10%
No	0.8766	—	—	—
BlurMe	0.8766	0.8686	0.8553	0.8385
BlurM(or)e	0.8766	0.8711	0.8640	0.8468

Table 11: The RMSE performance with Matrix Factorization on the original MovieLens dataset, BlurMe data and on BlurM(or)e data.

5.6 Generalizability

This section investigates whether the effects of gender inference and gender obfuscation generalize to other datasets. We repeat the experiments, but this time using the Flixster and LibimSeTi dataset, presented in section 3.1.

	Extra ratings			
Dataset	0%	1%	5%	10%
Original	0.67±0.08	—	—	—
BlurMe	0.67±0.08	0.22±0.08	0.07±0.03	0.03±0.02
BlurM(or)e	0.67±0.08	0.30±0.10	0.16±0.06	0.11±0.07

*Table 12: Gender inference performance on the **Flixster** dataset and on the obfuscated **Flixster** dataset. Performance measured in ROC AUC.*

	Extra ratings			
Dataset	0%	1%	5%	10%
Original	0.99±0.00	—	—	—
BlurMe	0.99±0.00	0.95±0.03	0.79±0.06	0.41±0.07
BlurM(or)e	0.99±0.00	0.94±0.05	0.83±0.14	0.65±0.29

*Table 13: Gender inference performance on the **LibimSeTi** dataset and on the obfuscated **LibimSeTi** dataset. Performance measured in ROC AUC.*

Table 12 and 13 reveal that gender inference and gender obfuscation is not exclusive to the MovieLens dataset. If we were able to use more than 400 users, we expect the gender inference attack to perform even better for the Flixster data, because we would have more data to train on. Gender inference in the LibimSeTi dataset is extremely easy. This might be due to the nature of the data: Male users will most likely rate female profiles and vice versa. The LibimSeTi dataset hence needs more obfuscation than 10% extra ratings. The fact that gender inference is possible on the LibimSeTi dataset proves that gender inference is also possible in other domains than movie-rating databases. This emphasizes the fact that gender inference is a universal phenomenon, which definitely needs attention.

5.7 Discussion - Obfuscation

This section has pointed to a weakness in a state-of-the-art gender obfuscation algorithm, BlurMe, and presents an improved algorithm, BlurM(or)e, that addresses the issue. With regard to the second research question, we have shown that BlurM(or)e is able to obfuscate gender in the user-item matrix without substantial utility loss of the dataset. We presented evidence that BlurM(or)e obfuscating is more difficult to identify by data visualization than BlurMe obfuscation.

Aside that, we have seen that the effects of gender inference and gender obfuscation are also present in other datasets. The results on the Flixster dataset show that gender inference is not only a threat to the MovieLens dataset. This is in line with the results of Weinsberg et al. (2012)[50], who showed that gender inference attacks are possible on the Flixster dataset. Even more so, the results on the dating dataset imply that the threat of gender inference is not exclusive to the domain of movie rating data, but also to other user-item interaction data. In fact, inference attacks seem to be possible on a variety of datasets. For instance, Salamatian et al. (2013) [41] presents an inference model that is capable of inferring political orientation (Republican vs. Democrat) from TV shows preferences. This underlines that inference attacks cannot only infer gender, but also other sensitive attributes such as political orientation.

Interestingly, BlurMe used the ROC area under the curve metric for the first gender inference experiments, yet changed to classification accuracy for the gender inference on the obfuscated dataset. Using accuracy as a performance metric on imbalanced datasets is a practice that should be avoided. It is advised to report the ROC AUC, precision-recall AUC and ROC AUC on skew-normalized data when dealing with imbalanced datasets (Jeni, Cohn and De La Torre, 2013 [19]).

Furthermore, BlurMe declares (in [50]) the classification accuracy of 2.5% as a success. However, in this scenario, one might argue that the attacking gender inference model should achieve a classification accuracy of 50%. With that score, it would perform as well as the random classifier in table 9, meaning that the inference model cannot distinguish between males and females. On the other hand, one could argue that gender would only be obfuscated if an attacking model predicts the wrong gender. In this case, achieving a classification accuracy close to zero would be optimal. Related literature

does not agree on this debate - some studies try to achieve an accuracy close to 0 (see Weinsberg et al., 2012 [50] or Reddy and Knight, 2016 [38]), and some studies aim to achieve an accuracy close to 50% (see e.g. Salamatian et al., 2013 [41]). Here, we chose to achieve a classification accuracy close to zero, in order to provide optimal gender obfuscation for each user in the dataset. With respect to the threat model (section 5.1), it would make more sense to obfuscate as strongly as possible. The inferred gender information of the newly encountered users becomes useless for a third party, since they are mostly wrong. However, our product does not prohibit an obfuscation until the 50% accuracy mark. One can e.g. just add $\approx 2\%$ extra ratings with BlurM(or)e to make the attacking model achieve an accuracy of about 50%. While that is also possible for the BlurMe obfuscation, the BlurM(or)e algorithm would still be more difficult to be detected. Thus, our proposed algorithm would still be superior to the BlurMe approach, even if we chose to obfuscate only until the 50% accuracy mark is reached.

One limitation of BlurM(or)e is that it might remove items from the dataset. As table 10 reveals, BlurM(or)e 10% has only 3699 items, while the original dataset consists of 3706 items. Due to the random removal of ratings, some movies might end up with no rating at all in the dataset, making them "disappear" in the user-item matrix. To solve this, we could restrict the removal to not remove ratings from movies with a low rating count.

Future work can address several problems; as mentioned before, normalization of the dataset can have an enormous impact on the classification performance. In table 3, we see that when our reproduction incorporates normalization the accuracy of gender inference still decreases with increasing obfuscation, but at a much slower rate. It is not clear why this phenomenon occurs.

Moreover, one could include more knowledge from part 1 in BlurM(or)e. For example, one could add ratings to certain movie genres instead of certain movies, making users more loyal to a certain movie genre. Alternatively, one could also experiment with the removal of ratings: BlurM(or)e currently removes random ratings from users with more than 200 ratings, however, it would be interesting to experiment with a smarter removal. For instance, one could remove ratings from movies that correlate with the own gender of a user. We expect that this would strengthen the obfuscation.

6 Part 3: Approaches for More Realistic Data

In the last section, we saw that a simple visualization reveals that the BlurMe algorithm was applied to the MovieLens dataset. As a response to that, we proposed an extension to BlurMe, that was robust against that flaw. In this chapter, we present a different approach to find out if a dataset is obfuscated or not. To do so, we train a classifier that will be able to distinguish between a real dataset and an obfuscated dataset. Our experiments show that the BlurMe as well as the BlurM(or)e data is easily distinguishable from the original data. If an attacker knows that a dataset is obfuscated, he might be able to reverse the obfuscation and thereby invade the privacy of the users in the dataset. The goal is thus to develop an obfuscation technique that obfuscates the gender of users in such a way that the resulting obfuscated dataset cannot be distinguished from an original, not-obfuscated dataset. We take two different approaches that address that problem: An adversarial approach that obfuscates the "fakeness" of the already obfuscated dataset, and one approach that keeps "realistic" data by obfuscating in a smarter way. In the remainder of this thesis, we refer to an original, not obfuscated dataset as a "real" dataset and to an obfuscated dataset as "fake" dataset.

To give a better impression of the practical relevance we take the browser extension "TrackMeNot" as a comparison: In 2006, AOL published search histories of their users. These queries are anonymized, meaning that there are no direct personal identifiers assigned to a search query. However, it was still possible to infer the identity of some users from the data, because the queries contained enough sensitive information about a user (e.g. queries involving geo-locations). To counter this privacy breach, Howe, Nissenbaum and Toubiana (2008) [17] developed "TrackMeNot", a browser extension that injects fake queries that protect the user from said inference attacks. The authors point out that the fake queries must be very realistic, otherwise it would be easy to remove these queries, which would leave the identity of the users unprotected.

Likewise, it is important in our scenario to make the fake data in the obfuscated dataset seem realistic, so that it is less likely that an attacker can successfully infer binary gender information.

6.1 General Threat model

Following our definition in section 2.4, we assume the following threat model for the experiments:

Resources: An attacker has access to a dataset, which he knows is real (for example the IMDB rating dataset) and has trained a gender inference model on it. He finds another dataset from the same domain, which he wants to attack (e.g. the MovieLens).

Goal: The attacker aims to find out if the MovieLens dataset is obfuscated or not by training a classifier that distinguishes between his dataset (IMDB) and the MovieLens dataset.

Vulnerability: If the classifier can distinguish well between the datasets (achieves a high classification accuracy), the attacker can assume that the MovieLens data is obfuscated. With this information, he can reverse the obfuscation by e.g. removing movies from the dataset that he believes are used for obfuscation. Subsequently, he can infer the gender of the users in the MovieLens dataset.

Countermeasure: The dataset should be gender-obfuscated in such a way that an attacker believes it is real. The attacker would thus infer the gender of obfuscated users

and achieve a low classification accuracy. The gender information of the users are thus be protected.

This threat model assumes that both datasets have the same movies, which is reasonable, since the attacker can use only movies that are present in both datasets.

6.2 Experiments - Real vs. Fake

By applying the described threat model on the BlurMe and BlurM(or)e dataset, we illustrate that a pretrained classifier can distinguish between obfuscated user profiles and not obfuscated profiles and thereby determine if a whole dataset is obfuscated or not. The classifier receives thus a user profile as input and outputs a prediction if the user is obfuscated or not. With the help of that classifier, the attacker can determine if a newly encountered dataset is obfuscated. The evidence underlines the need for realistic obfuscation.

Data: With respect to the threat model, the new dataset is constructed in the following way: Half of the dataset originates from the real data (e.g. the real MovieLens data). This half is further referred to as dataset "A". The other half of the dataset originates from the dataset that is assumed to be obfuscated (e.g. the BlurMe dataset), referred to as "B". In our experiments, we take the first 3020 users from the original MovieLens dataset and the last 3020 users from the obfuscated MovieLens dataset. In this way, one can be certain that no user appears in the new dataset twice. Users from the original dataset are labeled as "real", while users from the obfuscated dataset are labeled as "fake". The resulting dataset will be used as training and testing data in the experiments. Figure 7 visualizes the process of creating the training and testing data.

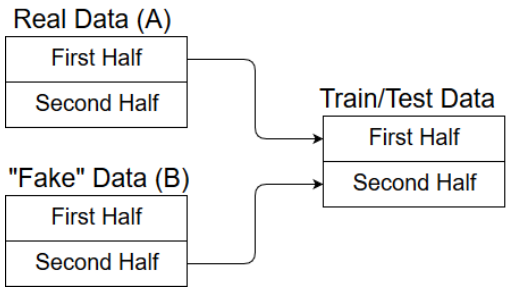

Figure 7: The creation of the new dataset for the Real vs. Fake experiments visualized. The "Real Data (A)" is the dataset that the attacker possesses and knows that it is real. The "Fake Data (B)" is the dataset that the attacker wants to attack.

Experimental Setup & Results: We use different basic machine learning models to distinguish between real and fake data. Analogue to the first experiments of gender inference in section 3.2, we use a SVM, logistic regression, multinomial Bayes and Bernoulli Bayes classifier with the same parameter settings. We will henceforth refer to the classifier that tries to distinguish between real and obfuscated data as "Real vs. Fake classifier". The Real vs. Fake classifiers will distinguish between the original MovieLens data and the BlurMe or BlurM(or)e obfuscated dataset respectively. The BlurMe and BlurM(or)e datasets are created with 10% extra ratings and the greedy strategy, because it yields the best gender obfuscation and the lowest RMSE respectively for each algorithm. Table 14 displays the performances of the different classifiers. The performance is measured in average classification accuracy over tenfold cross-validation. This

choice is justified because we have a balanced dataset: there are as many real users as obfuscated users in the dataset (Jeni, Cohn and De La Torre, 2013 [19]).

The second column of table 14, "Real vs. Real" serves as baseline classification accuracy. In this setting, the classifiers were supposed to distinguish between the first half and the second half of the original MovieLens data. This is a good baseline, because both halves are part of the original dataset. That setting thus reflects the scenario in which an attacker tries to determine if the dataset B is obfuscated, but it is actually real. If the attacker would achieve an accuracy of 68%, he would conclude that the dataset B is not obfuscated. Any dataset B that reaches an accuracy score of around 68% will be considered realistic and any dataset B that reaches an accuracy of around 1 will be considered obfuscated and obviously artificially created.

Classifier	Real vs. Real	Real vs. BlurMe	Real vs. BlurM(or)e
Random	0.50±0.00	0.50±0.00	0.50±0.00
Bernoulli	0.63±0.03	0.762±0.021	0.757±0.041
Multinomial	0.68±0.06	0.965±0.006	0.887±0.058
SVM	0.67±0.05	0.995 ± 0.003	0.972±0.019
Logistic	0.68±0.05	0.995±0.004	0.977±0.018

Table 14: Real vs. Fake classification results using different machine learning models measures in classification accuracy over 10-fold cross-validation. BlurMe and BlurM(or)e both use the greedy strategy with 10% extra ratings on the MovieLens dataset.

Interestingly, every classifier beats their respective baseline by far. That means on the one hand, that it is possible to train a classifier that can reliably detect obfuscated data. On the other hand, one can clearly see that the obfuscated datasets (BlurMe and BlurM(or)e) must seem very artificial. BlurM(or)e seems to look a bit less obfuscated than BlurMe.

Since logistic regression performs the best and offers great insights in feature importance, we will use that model for the rest of the experiments.

6.3 Toward More Realistic Data

The last section showed that the BlurMe and the BlurM(or)e approaches create artificially appearing datasets that are immune to gender inference attacks. However, if an attacker knows that a dataset is obfuscated, he might successfully reverse the obfuscation and thereby breach the users' privacy (like we showed in section 5.2). The need for realistic appearing data that blocks gender inference is obvious.

We propose two different approaches to make the dataset appear more realistic:

Approach 1: Adversarial Re-applying BlurM(or)e: The field of adversarial machine learning deals with machine learning techniques against an adversarial opponent (Huang, Joseph, Nelson, Rubinstein and Tygar, 2011 [18]; Kurakin, Goodfellow and Bengio, 2016 [23]). The goal is to fool the adversary with malicious input. For example, SpamBayes is an e-mail spam detection model that detects spam e-mails among regular e-mails. It calculates a "spam-score" for an e-mail by looking at spam indicative words within the e-mail. This model can be fooled by avoiding those spam indicative words and use less indicative words with similar meaning instead (see Wittel and Wu, 2004 [52], and Lowd and Meek, 2005 [29]).

In our context, we aim to fool the Real vs. Fake classifier. We would like to achieve an accuracy of about 68% with an obfuscated dataset retaining gender inference blocking. To achieve that, we apply a variation of the BlurM(or)e algorithm once again on the

obfuscated dataset, but this time, the algorithm obfuscates the "fakeness" instead of the gender.

We thus start by training a logistic regression model on the dataset that contains real users and obfuscated users. The most correlating movies for real data and obfuscated data are extracted into two ordered lists, called L_r (the list with movies that correlate with realistic users) and L_o (the list with movies that correlate with obfuscated users). To make the gender-obfuscated dataset look more realistic, we add ratings for movies that correlate with "realness" to every user in the dataset. Similar to BlurMe and BlurM(or)e, we investigate 1%, 5% or 10% extra ratings. After adding the ratings, we remove ratings from user profiles by applying the greedy strategy with list of movies that correlate most with obfuscated users - making the obfuscated dataset seem more realistic. We remove as many ratings per users as we added beforehand, so that the number of interactions in the dataset stays constant. Figure 8 visualizes the process for further understanding.

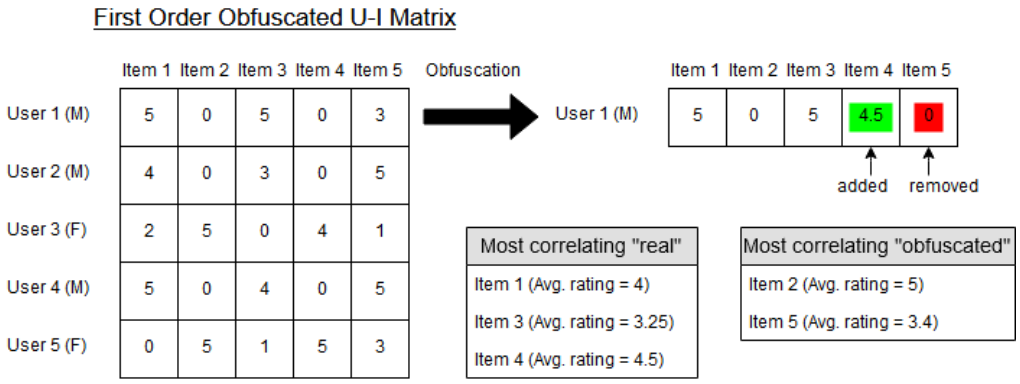

Figure 8: The adversarial approach visualized: User 1 is obfuscated (again) by adding a rating to item 4, because it correlates with "realness" and removing the rating from item 5, since it correlates with "obfuscated".

The movies that correlate the most with obfuscated users (movies from L_o) in the BlurMe dataset are also in the lists of movies that correlate with gender (see table 6). This was to be expected, since we applied obfuscation by adding ratings to those movies. It is expected that after the removal of ratings from L_o, gender inference is possible again. Surprisingly, the lists of most correlating movies from the BlurM(or)e dataset is quite different from the other lists. Table 15 and table 16 show the lists L_r and L_o of the BlurMe and BlurM(or)e dataset respectively.

Rank	L_r	L_o
1	Almost Famous	Steel Magnolias
2	Dog Day Afternoon	Grand Hotel
3	Return to Me	Go West
4	One Flew Over the Cuckoo's Nest	Carrie
5	Meet the Parents	Simply Irresistible

*Table 15: Top 5 movies from the lists L_r and L_o for the MovieLens dataset **after BlurMe** obfuscation was applied with the greedy strategy and 10% extra ratings.*

Rank	L_r	L_o
1	Meet the Parents	Romancing the Stone
2	Pretty in Pink	Back to the Future Part II
3	Remember the Titans	Dumb & Dumber
4	Lost World: Jurassic Park	Sex, Lies, and Videotape
5	Fatal Attraction	Platoon

Table 16: Top 5 movies from the lists L_r and L_o for the MovieLens dataset ***after BlurM(or)e*** *obfuscation was applied with the greedy strategy and 10% extra ratings.*

To test if the second obfuscation works, we set up the experiments like in section 3.2. We train a logistic regression model on the dataset that consists of the real data and the first order obfuscated MovieLens data and test that model on the data that consists of the real and second order obfuscated MovieLens data. The first-order obfuscated MovieLens data results from applying the gender-obfuscation algorithms BlurMe or BlurM(or)e to the MovieLens data. The second-order obfuscated data results from the adversarial approach, which obfuscates the "fakeness" of the first order obfuscated dataset. Table 17 displays that comparison. The performance is measured in classification accuracy. As a baseline, we chose to use a random classifier, which has no knowledge about anything. It will classify a user as "obfuscated" with a probability of 50%, otherwise it will classify that user as "real".

			extra ratings		
Classifier	First-order obf.	second-order obf.	1%	5%	10%
Log. Reg.	BlurMe	BlurM(or)e extension	0.52±0.01	0.51±0.01	0.51±0.01
Log. Reg.	BlurM(or)e	BlurM(or)e extension	0.54±0.02	0.51±0.01	0.51±0.01
Random	BlurMe	BlurM(or)e extension	0.50±0.01	0.50±0.01	0.50±0.01
Random	BlurM(or)e	BlurM(or)e extension	0.50±0.01	0.50±0.01	0.50±0.01

Table 17: Performance of the adversarial approach measured in classification accuracy of a logistic regression model (Log. Reg.) and a random classifier (Random). All dataset were created with the greedy strategy on the MovieLens dataset.

		Ground Truth	
		Fake	Real
Prediction	Fake	2	6
	Real	272	268

Table 18: Confusion matrix of one fold of the logistic regression model used in the experiments. One can see that the classifier is successfully fooled and classifies almost every user as "Real".

Table 17 reveals that the Real vs. Fake classifier cannot distinguish between real and obfuscated users. It cannot systematically outperform the random classifier and has thus no knowledge about real or obfuscated users. Looking at the confusion matrix (Table 18), we find that the logistic regression model classifies (almost) every user as "real". This confirms the "fakeness" obfuscation, because no user is considered to be fake. However, this approach assumes that the attacker has a model that was trained to distinguish between the real and BlurMe or BlurM(or)e data with 10% extra ratings added via the greedy strategy. Assuming the attacker has these resources ("resources" as term of the threat model, cf. section 6.1), the second-order obfuscation approach

would be valid. Since the attacker would have to have a pretrained model for all the parameter settings, this assumption of resources is not realistic. It is more likely that the attacker trains his model again as soon as he receives a new dataset. Using this attacking strategy, the attacker can still find out if the dataset is obfuscated or not. We thus train a new logistic regression model, but this time it tries to distinguish between the original (real) data and the second order obfuscated BlurMe or BlurM(or)e data and report the results in table 19.

			extra ratings		
Classifier	First order obf.	second order obf.	1%	5%	10%
Log. Reg.	BlurMe	BlurM(or)e extension	0.96±0.01	0.99±0.01	0.99±0.00
Log. Reg.	BlurM(or)e	BlurM(or)e extension	0.96±0.02	0.97±0.02	0.98±0.02

Table 19: Performance of a logistic regression classifier that tries to distinguish between the real MovieLens data and the second order obfuscated MovieLens data measured in terms of classification accuracy. All obfuscated datasets are created with the greedy strategy.

Table 19 shows that the attacker is still able to determine if a dataset is obfuscated or not by simply training his model again. This attacking strategy is thus far superior and requires less resources. One could go on with a third order obfuscation by applying the BlurM(or)e extension all over again, but we doubt that this is the right way to counter the attack. Also, it would heavily change the data and that would have an enormous impact on the data quality. To our knowledge, no one has addressed this topic, so future research might investigate that.

Thus, we present another, non-adversarial approach.

Approach 2: Smarter obfuscation: As one can see in table 14, the classifier is able to distinguish between the real and BlurMe or BlurM(or)e data extremely well. This might be due to the fact that we add ratings to every single user in the user-item matrix. However, it is not necessary to add ratings to every user - it would make more sense to only add ratings to users that are easy to classify (or rather users whose gender can be easily inferred). Thus, we propose another approach which employs this idea. The new algorithm, for short called BlurMeBetter, assigns every user a score that represents the prediction confidence of a gender inference model. Based on that score, the user will be obfuscated with the BlurM(or)e approach or not. With that method, we obfuscate smarter and keep the data more realistic.

The algorithm in detail works like this: We start, yet again, by training a logistic regression model on the original MovieLens dataset. We extract the two lists, L_f and L_m, that correlate with either gender, exactly like BlurMe and BlurM(or)e. Also, the algorithm memorizes how certain the logistic regression model is on the users in the test set. Since we are using cross-validation, every user is present in the test set once. A logistic regression model provides insight about how confident the model is about a single user (i.e., a certain user is with a probability of 95% make and 5% female, we take the maximum of these probabilities as certainty value, so 95%). That certainty value is scaled from 0 to 1, where 1 means that the model is extremely certain about the gender of one user. If the model misclassifies one user, the certainty is set to 0, regardless how certain the model actually was. With both lists and a certainty score for every user, we iterate over all users to obfuscate them: If the user has a certainty score smaller than a threshold $\mathbf{c}$, (s)he will not be obfuscated. If (s)he does have a certainty score higher than $\mathbf{c}$, we apply the BlurM(or)e obfuscation algorithm to that user.

The parameter **c** influences the number of skipped users (the number of not obfuscated users). The higher the **c**-value, the more users will be skipped. In return, the dataset as a whole will appear to be more realistic. However, if too many users are skipped, gender inference will be possible again, implying that **c** should be sufficiently low to prevent successful attacks on obfuscated data. There are no studies to our knowledge that investigate a good threshold for the certainty value, so we will find one empirically.

Hence, we obfuscate smarter, by only obfuscating users that really need obfuscation. We test that obfuscation algorithm by reproducing the experiments of section 5.4. The performance of the Real vs. Fake classifier is compared to the performance of a Real vs. Real classifier. The Real vs. Real classifier is a logistic regression model that tries to distinguish between real users and real users. For this classifier, we use the whole the MovieLens dataset, but label 50% of the users as "obfuscated". This setting reflects the scenario where the attacker finds a non-obfuscated dataset and tries to find out if that dataset is obfuscated or not, just like column 2 in table 14. Of course, we want the attacker to think that our obfuscated dataset is real when he finds our dataset. Thus a realistic appearing obfuscated dataset should result in a performance similar to a real dataset. If it does, one can consider that dataset is robust against the "Real vs. Fake" attack. Table 20 shows the robustness of gender inference attacks after BlurMeBetter is applied, while table 21 displays the robustness against the Real vs. Fake classifier.

			extra ratings		
dataset	**c**-value	#skipped users	1%	5%	10%
BlurMe	—	—	0.54±0.03	0.15±0.03	0.02±0.01
BlurM(or)e	—	—	0.64±0.03	0.36±0.07	0.19±0.07
BlurMeBetter	c=0.5	1677	0.67±0.03	0.34±0.07	0.07±0.03
BlurMeBetter	c=0.8	1948	0.68±0.03	0.37±0.07	0.10±0.04
BlurMeBetter	c=0.95	2305	0.70±0.03	0.41±0.06	0.13±0.03
BlurMeBetter	c=0.99	2756	0.73±0.02	0.47±0.06	0.19±0.04

Table 20: Gender inference on the MovieLens dataset obfuscated by BlurMeBetter for different c-values.

			extra ratings		
Real vs. ...	**c**-value	#skipped users	1%	5%	10%
Real (baseline)	—	—	0.68±0.05		
BlurMe	—	—	0.97±0.01	0.99±0.01	0.99±0.01
BlurM(or)e	–	—	0.87±0.03	0.94±0.04	0.98±0.02
BlurMeBetter	0.5	1677	0.79±0.05	0.83±0.04	0.85±0.03
BlurMeBetter	0.8	1948	0.78±0.05	0.81±0.04	0.84±0.04
BlurMeBetter	0.95	2305	0.77±0.05	0.81±0.04	0.82±0.04
BlurMeBetter	0.99	2756	0.75±0.05	0.78±0.04	0.79±0.04

Table 21: Performance of the Real vs. Fake logistic regression classifier on different datasets measured in classification accuracy. All datasets were created using the greedy strategy.

Table 20 confirms that BlurMeBetter is capable of gender obfuscation, at least if one adds more than 5% or 10% extra ratings. Interestingly, BlurMeBetter outperforms the BlurM(or)e algorithm in terms of gender obfuscation. Moreover, table 21 shows

that using the BlurMeBetter obfuscation results in more realistic data than using the BlurMe or BlurM(or)e algorithm. Thus, BlurMeBetter is superior to BlurM(or)e in both aspects: Gender inference blocking and realistic looking data.

Of course there is a trade-off between realistic data and gender inference blocking when using the BlurMeBetter algorithm. For high **c**-values, fewer users are obfuscated, making the dataset appear real. However, obfuscating only a few users makes gender inference easier. For the MovieLens data, a c-value of 0.99 skips almost half of the users. Table 20 shows that obfuscating only around half of the dataset is already enough to block gender inference (for $c = 0.99$ we skip 2756 users out of all 6040 users).

The best performing parameter setting is a **c**-value of 0.99 and 10% extra ratings, because the performance of the Real vs. Fake classifier for that setting is close to the baseline performance of the Real vs. Real classifier, while maintaining the gender inference blocking. Note that with this setting, there is still a difference of 11% to the baseline (Real vs. Real) classifier. Taking the standard deviations of the accuracies into account, we believe that this difference is not substantial enough for an attacker to safely assume that the "fake" dataset is obfuscated.

6.4 Quality of the Dataset

We compare the data quality of the BlurMeBetter approach to the BlurMe and BlurM(or)e approaches in terms of RMSE. It is expected that the RMSE for the BlurMeBetter approach is closer than BlurMe or BlurM(or)e to the original dataset, because we alter the data less. More detailed, we compare BlurMe, BlurM(or)e and BlurMeBetter that all employ 1%, 5% or 10% extra ratings with the greedy strategy in table 22. For BlurMeBetter, we use a **c**-value of 0.99.

	Extra ratings			
Obfuscation	0%	1%	5%	10%
Original	0.8766	—	—	—
BlurMe	0.8766	0.8686	0.8553	0.8385
BlurM(or)e	0.8766	0.8711	0.8640	0.8468
BlurMeBetter	0.8766	0.8739	0.8665	0.8524

Table 22: The RMSE performance with Matrix Factorization on the original data, BlurMe data, BlurM(or)e data and BlurMeBetter data. All datasets employ the greedy strategy, BlurMeBetter uses a ***c****-value of 0.99.*

As expected, the RMSE of the BlurMeBetter dataset is closest to the RMSE of the original dataset in all categories. The gender obfuscation thus leaves the utility of the dataset intact.

6.5 Generalizability

To test if the effects of the Real vs. Fake classifier and the BlurMeBetter approach also generalize to other datasets, we repeat the same experiments with the Flixster and LibimSeTi dataset. As mentioned above in section 3.1, it is only possible to take a subset of 400 users for both datasets. Nonetheless, the effects of gender blocking and realistic obfuscation should be present in the subsets. Table 23 and table 24 show the results of the Real vs. Fake classifier and gender inference attacks on the Flixster data, while table 25 and table 26 show the results on the LibimSeTi dataset.

	Extra ratings			
Real vs.	0%	1%	5%	10%
Original	0.54±0.06	—	—	—
BlurMe	0.54±0.06	0.89±0.06	0.88±0.05	0.88±0.03
BlurM(or)e	0.54±0.06	0.74±0.09	0.75±0.09	0.80±0.06
BlurMeBetter	0.54±0.06	0.55±0.06	0.54±0.05	0.54±0.06

*Table 23: Performance of the Real vs. Fake classifier on the **Flixster** dataset, obfuscated with different strategies. Performance measured in classification accuracy.*

	Extra ratings			
Dataset	0%	1%	5%	10%
Original	0.67±0.08	—	—	—
BlurMe	0.67±0.08	0.22±0.08	0.07±0.03	0.03±0.02
BlurM(or)e	0.67±0.08	0.30±0.10	0.16±0.06	0.11±0.07
BlurMeBetter	0.67±0.08	0.59±0.08	0.47±0.11	0.44±0.12

*Table 24: Gender inference performance on the **Flixster** dataset and on the obfuscated **Flixster** dataset. Performance measured in ROC AUC.*

	Extra ratings			
Real vs.	0%	1%	5%	10%
Original	0.52±0.09	—	—	—
BlurMe	0.52±0.09	0.73±0.06	0.74±0.06	0.75±0.09
BlurM(or)e	0.52±0.09	0.66±0.09	0.70±0.09	0.73±0.09
BlurMeBetter	0.52±0.09	0.59±0.07	0.59±0.08	0.64±0.09

*Table 25: Performance of the Real vs. Fake classifier on the **LibimSeTi** dataset, obfuscated with different strategies. Performance measured in classification accuracy.*

	Extra ratings			
Dataset	0%	1%	5%	10%
Original	0.99±0.00	—	—	—
BlurMe	0.99±0.00	0.95±0.03	0.79±0.06	0.41±0.07
BlurM(or)e	0.99±0.00	0.94±0.05	0.83±0.14	0.65±0.29
BlurMeBetter	0.99±0.00	0.99±0.01	0.91±0.06	0.76±0.16

*Table 26: Gender inference performance on the **LibimSeTi** dataset and on the obfuscated **LibimSeTi** dataset. Performance measured in ROC AUC.*

The tables confirm the effects found for the MovieLens data and thereby providing support for the representativeness of the dataset and broader usability of our proposed algorithm in practice. BlurMeBetter is capable of obfuscating the gender of users, while keeping the dataset realistic. Since gender inference is very easy in the LibimSeTi dataset, it needs more obfuscation. We expect better gender obfuscation for more extra ratings, but at the cost of realistic appearance.

In general, both datasets, Flixster and LibimSeTi, confirm the fact that all three obfuscation approaches are able to obfuscate gender in a user-item matrix. BlurMeBetter, however, results in the most realistic appearing data, no matter how many fake ratings are added to the data.

6.6 Discussion - Realistic data

This section extends the previous section by using a more complex approach to identify an obfuscated dataset. Instead of exploiting simple visualization of movie-rating frequencies, this section presents basic machine learning classifiers that are able to reliably identify obfuscated users. With these models, an attacker is able to identify obfuscated datasets. Assuming that the attacker is able to reverse obfuscation, these classifiers pose a substantial threat to user privacy. It is thus necessary to obfuscate a dataset without making it appear obfuscated.

This chapter contributes to the third research question "How can we obfuscate the gender of a dataset while maintaining the dataset quality and a realistic appearance?" by proposing two different approaches: An adversarial approach, that successfully obfuscates the "fakeness" of an already gender-obfuscated dataset, which successfully "fools" a pretrained Real vs. Fake classifier. The second approach, however, does not assume that an attacker uses a pretrained classifier We presented BlurMeBetter, which obfuscates only selected users, who actually "need" obfuscation. This approach shows more resistance to the Real vs. Fake classifier, meaning that it is more difficult for an attacker to determine if the newly encountered dataset is obfuscated or not. Also, BlurMeBetter is robust against gender inference attacks.

The generalization analysis reveals that these properties also hold for the Flixster dataset and the LibimSeTi dataset. One can assume that the BlurMeBetter approach can be applied for gender obfuscation for any user-item matrix.

However, there are some points that need to be discussed: Firstly, the attacker determines if a dataset is obfuscated by distinguishing the presumably obfuscated user profiles from real user profiles. This attack does not use global statistics of the obfuscated dataset, like rating density, average rating per movie etc. It is possible that these global statistics help identifying obfuscated datasets. It would make sense from the attacker perspective to include these as features for the classifier.

Secondly, the generalizability analysis is fairly limited, because we only used a small fraction of users (400) of the Flixster and LibimSeTi dataset. Due to limitations of computational power, it was not possible to increase the subset. Nonetheless, the generalizability study evidences that the results found in this chapter also generalize to other datasets.

Moreover, this study investigates only basic machine learning classifiers as real vs. fake classifier (see table 14). Maybe, similar to the approach of Liu, Qu and Mahmud (2019) [28] for gender inference, there might be a more complex approach for distinguishing between obfuscated and not obfuscated user profiles. For example, one could develop a classifier involving deep learning, which generally showed incredible success during the last few years and achieving state-of-the-art performances across several fields, like speech recognition, visual object recognition or object detection (LeCun, Bengio and Hinton, 2015 [26]).

Lastly, this study assumes that the rating behavior of users is similar across platforms. However, it is possible that the users from the MovieLens dataset rate differently than the users in the IMDB or Netflix data. Meo, Ferrara, Abel, Aroyo and Houben (2013)[31], found that users seem to behave differently across social sharing platforms (Flickr, Delicious and StumbleUpon). While these platforms are from another domain, similar effects might occure in the domain of movie rating data. This study uses only users from the MovieLens dataset for simplicity, but future research should definitely investigate if that assumption is valid.

7 Overall Discussion and Conclusion

This study contributes to gender obfuscation in a user-item matrix to protect users' private and sensitive information. We showed that a gender inference attack on movie rating data is relatively easy. Our gender inference model achieved similar success as Weinsberg et al. (2012) [50] and Liu, Qu, Chen and Mahmud (2009) [28]. A countermeasure that blocks gender inference attacks, however, proves to be more difficult. Weinsberg et al. (2012) [50] presented a gender obfuscation algorithm, which served as basis for our developed algorithms. We pointed toward a flaw in the BlurMe algorithm - it is possible to determine by simple data visualization (or more complex machine learning approaches) that BlurMe has been applied to a dataset. Once an attacker knows that BlurMe has been applied, he can reverse the obfuscation, leaving the privacy of the users unprotected. Our proposed obfuscation algorithms, BlurM(or)e and BlurMeBetter both prove to obfuscate gender in the user-item matrix, while being more difficult to determine if any obfuscation has been applied. We recommend BlurMeBetter with the **c**-parameter of 0.99, because it successfully obfuscates the users' gender, while appearing very similar to the original dataset. BlurMeBetter is a novel approach that contributes to more privacy protection in user-item interaction data. With the help of BlurMeBetter, the "organization" privacy of users is protected, making the users feel more secure while interacting with recommender systems.

Despite various robustness checks, some of the results we obtained in this study may be sensitive to our methodological choices. First of all, this study investigates adding and removing ratings from an original dataset to obfuscate gender information. While this proves to be very effective, we are changing the ground truth of the dataset. The original dataset was created by observing real user-item interactions. The obfuscated dataset, however, also contains fictive ratings, that that do not originate from real user interaction. The added ratings might not reflect the original users' rating behavior, making the new dataset less usable to train and validate recommender systems. However, we showed in section 5.5 and section 6.4 that the utility of the obfuscated datasets are kept in terms of RMSE. Aside that, the movies that are used for adding extra ratings cannot be used for recommendation anymore, because they are already rated. This might be a reason why the RMSE decreases in our experiments (see table 11 and 22).

During this paper, we mention different attacks that identify obfuscated dataset. These attacks are only hypothetical, because we do not prove that they succeed in a real world scenario. For example, we show that data simple visualization can reveal that the BlurMe algorithm has been applied, because some movies have an unrealistically high rating count. In fact, we compare these statistics between the original and obfuscated MovieLens dataset, but a real world attacker would not have access to the original dataset. The attacker would thus have to compare the statistics of the obfuscated MovieLens dataset with the statistics from e.g. the IMDB. We are extremely certain that this attack would work, but leave it to future work to validate this.

Another limitation of this study is that the original code of the BlurMe algorithm was not available. The algorithm had to be re-implemented from scratch and it was possible to achieve very close, but not identical results as in Weinsberg et al. (2012) [50]. Nonetheless, our re-implementation showed the same effects as the original, so the comparisons of the obfuscation algorithms are valid. To prevent this problem for future research, we decided to make all code used in this study publicly available at `https://github.com/STrucks/BlurMore`.

This study promotes a strong obfuscation, i.e. reducing the gender inference accuracy as much as possible. This offers strong protection towards the users in the dataset, but it also implies a risk: As soon as an attacker finds out that a dataset is strongly gender-obfuscated, he can reverse the obfuscation by applying his pretrained gender inference

model and flip the predictions. If the model predicts for a certain user profile a male gender, the attacker can assume that the real gender of the user is female. This poses a risk of strong obfuscation and should be taken into account in deciding the operational settings (i.e. how many extra ratings) for the BlurM(or)e and BlurMeBetter algorithm. While both algorithms, BlurM(or)e and BlurMeBetter are designed to prevent such an attack, an attacker might find a way to identify if one of these obfuscation algorithms have been applied.

This study provides a great foundation for future research in this field, exploring different extensions and variations of the proposed algorithms. For example, we saw in Part 1 (section 4) that most movies contain gender indicative information. Hence, not using 2000 of the most important movies still results in a reasonable AUC of the gender inference model. Assuming that the obfuscation algorithm only uses movies that are important (e.g. the greedy strategy), it would be possible for an attacker to ignore these movies, making gender inference possible again. This hypothetical attack works if the obfuscation algorithm uses only the most important movies. The random strategy, however, would be immune to that attack, because it adds ratings to random and possibly irrelevant movies. Unfortunately, the random strategy does a poor job in obfuscating the gender. Nonetheless, future work might find a way which adds ratings to a larger variety of movies so that the gender is obfuscated and the hypothetical attack is prevented. One possible solution would be to use different parameters for the BlurM(or)e algorithm. The BlurM(or)e algorithm removes movies from the lists of most correlating movies once they doubled their original rating count. Here we used the factor 2, i.e. doubled, but if we would lower that factor, e.g. 1.5, a movie would be removed faster from the lists. In return, a larger variety of movies will be used for gender obfuscation.

Furthermore, it would be interesting to apply the obfuscation approaches to attributes. One might argue that gender is not very sensitive and one should rather obfuscate sexual or political orientation. Other work has shown that it is possible to infer political orientation from a user-item matrix (see Salamatian et al. (2013) [41]). Future research could investigate if it is also possible to obfuscate these, arguable more sensitive attributes by applying the BlurMeBetter algorithm.

Moreover, we discovered that normalization of the user-item matrix has a substantial effect on gender inference. Unfortunately, we cannot find a reason for this effect, or find any other studies that discover this effect as well. Future research can thus investigate this.

In part 3 (section 6), we presented an adversarial approach to generate realistic appearing data. Although that approach performed substantially worse than BlurMeBetter, future research could extend this idea. For example, one could train a generative-adversarial-network (GAN) which produces realistic appearing gender-obfuscated data. Triastcyn and Faltings (2018), for instance, showed that GANs are able to generate artificial, privacy-preserving high-dimensional datasets. Using this approach to obfuscate a user-item matrix opens up ways for synthetic, privacy-preserving data. Of course, the privacy of a dataset is maximized if it only contains synthetic i.e. hypothetical data.

In addition, the BlurM(or)e approach, which is the basis of the BlurMeBetter algorithm, removes ratings in a non-optimal way. In fact, it removes ratings from users that have more than 200 ratings. This choice was motivated by the assumption that we most likely do not hurt the obfuscation of these users, because they already provide so much data. However, different removal strategies might even enhance the obfuscation. For instance, one could remove ratings from movies that correlate with the gender of the respective user. With that strategy, we would add ratings to movies that correlate with the user's opposite gender and remove ratings from movies that correlate with the user's gender. If we remove as many ratings per user as we add, the number of interactions would stay constant. Nonetheless, we would heavily change each user profile and that

must have a big impact on the data quality. It would still be interesting to see how variations of the obfuscation algorithms proposed in this work would perform in terms of gender obfuscation and maintenance of data quality.

Aside that, one could also vary on the strength of obfuscation while obfuscating a dataset. For example, the algorithm adds randomly 1%, 5% or 10% extra ratings to a user's profile. This would make reversing the obfuscation more difficult. To make it even more difficult, one could also vary on the obfuscation strategy (random, sampled or greedy strategy).

In Appendix A, we present a failure analysis for the gender inference model from section 3.2. It reveals that users who rated a high number of movies are easier to classify than users with a few number of ratings. Also, users who are more loyal to a movie genre seem to be more difficult to classify. Future work might incorporate this knowledge to build even more effective gender obfuscation algorithms.

Finally, one could extend the generalizability analysis of this study. Due to computational constrains, the analysis was restricted to 400 users from the Flixster and LibimSeTi dataset. However, future work could apply the proposed algorithms to datasets from other domains (e.g. Book ratings, Facebook Likes, ...).

References

[1] Data breach quick view report 2016 data breach trends–year in review. https://pages.riskbasedsecurity.com/hubfs/Reports/2016 Year End Data Breach QuickView Report.pdf, 2017. Accessed: 2019-05-29.

[2] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[3] Shahriar Badsha, Xun Yi, and Ibrahim Khalil. A practical privacy-preserving recommender system. *Data Science and Engineering*, 1(3):161–177, 2016.

[4] Shahriar Badsha, Xun Yi, Ibrahim Khalil, and Elisa Bertino. Privacy preserving user-based recommender system. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1074–1083. IEEE, 2017.

[5] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SiGKDD Explorations*, 9(2):75–79, 2007.

[6] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Jesús Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[7] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[8] Lukas Brozovsky and Vaclav Petricek. Recommender system for online dating service. In *Proceedings of Znalosti 2007 Conference*, Ostrava, 2007. VSB.

[9] Jian Cao, Hengkui Hu, Tianyan Luo, Jia Wang, May Huang, Karl Wang, Zhonghai Wu, and Xing Zhang. Distributed design and implementation of svd++ algorithm for e-commerce personalized recommender system. In *13th National Conference on Embedded System Technology*, pages 30–44. Springer, 2015.

[10] Terence Craig and Mary E Ludloff. *Privacy and Big Data: the Players, Regulators, and Stakeholders.* " O'Reilly Media, Inc.", 2011.

[11] Li Deng and Y Dong. Foundations and trends® in signal processing. *Signal Processing*, 7:3–4, 2014.

[12] Nathaniel I Durlach, Christine R Mason, Gerald Kidd Jr, Tanya L Arbogast, H Steven Colburn, and Barbara G Shinn-Cunningham. Note on informational masking (l). *The Journal of the Acoustical Society of America*, 113(6):2984–2987, 2003.

[13] Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

[14] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L Lagendijk. Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE transactions on information forensics and security*, 7(3):1053–1066, 2012.

[15] Olivier Gouvert, Thomas Oberlin, and Cédric Févotte. Matrix co-factorization for cold-start recommendation. In *ISMIR*, pages 792–798, 2018.

[16] J. Hassler. The power of personalized product recommendations. https://www.intelliverse.com/blog/the-power-of-personalized-product-recommendations/, 2017. Last updated: 22 Aug 2017; Accessed: 2019-06-25.

[17] Daniel C Howe, Helen Nissenbaum, and Vincent Toubiana. Trackmenot. *mrl. nyu. edu/dhower/trackmenot*, 2008.

[18] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[19] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 245–251. IEEE, 2013.

[20] Yancheng Jia, Changhua Zhang, Qinghua Lu, and Peng Wang. Users' brands preference based on svd++ in recommender systems. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, pages 1175–1178. IEEE, 2014.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer Society Press*, 42(8):30–37, 2009.

[22] John Krumm. Inference attacks on location tracks. In *International Conference on Pervasive Computing*, pages 127–143. Springer, 2007.

[23] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[24] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008.

[25] Martha Larson, Alessandro Zito, Babak Loni, and Paolo Cremonesi. Towards minimal necessary data: The case for analyzing training data requirements of recommender algorithms. *In Proceedings of Workshop on Responsible Recommendation at ACM RecSys'17, Como, Italy, 31 August 2017 (FATREC'17), 6 pages*, 2017.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[27] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[28] Yongsheng Liu, Hong Qu, Wenyu Chen, and SM Hasan Mahmud. An efficient deep learning model to infer user demographic information from ratings. *IEEE Access*, 7:53125–53135, 2019.

[29] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *CEAS*, volume 2005, 2005.

[30] Ian MacKenzie, Chris Meyer, and Steve Noble. How retailers can keep up with consumers. *McKinsey & Company*, 2013.

[31] Pasquale de Meo, Emilio Ferrara, Fabian Abel, Lora Aroyo, and Geert-Jan Houben. Analyzing user behavior across social sharing environments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):14, 2013.

[32] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[33] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 ieee symposium on security and privacy*, pages 111–125. IEEE, 2008.

[34] Mehrbakhsh Nilashi, Othman Ibrahim, and Karamollah Bagherifard. A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. *Expert Systems with Applications*, 92:507–520, 2018.

[35] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE, 2008.

[36] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Sravana Reddy and Kevin Knight. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 17–26, 2016.

[39] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–59, 1997.

[40] Stacey Ronaghan. The mathematics of decision trees, random forest and feature importance in scikit-learn and spark. https://medium.com/@srnghn/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3, 2018. Accessed: 2019-05-07.

[41] Salman Salamatian, Amy Zhang, Flavio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. How to hide the elephant-or the donkey-in the room: Practical privacy against statistical inference for large data. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 269–272. IEEE, 2013.

[42] Chris Salter, O Sami Saydjari, Bruce Schneier, and Jim Wallner. Toward a secure system engineering methodolgy. In *Proceedings of the 1998 workshop on New security paradigms*, pages 2–10. ACM, 1998.

[43] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[44] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[45] Cheryl Tang. Top 3 reasons to include data masking in your data security strategy. https://www.imperva.com/blog/top-3-reasons-include-data-masking-data-security-strategy/, 2017. Accessed: 2019-05-29.

[46] Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame. Data sharing by scientists: Practices and perceptions. *PloS one*, 6(6):e21101, 2011.

[47] Koen Verstrepen, Kanishka Bhaduriy, Boris Cule, and Bart Goethals. Collaborative filtering for binary, positive only data. *ACM SIGKDD Explorations Newsletter*, 19(1):1–21, 2017.

[48] Cong Wang, Yifeng Zheng, Jinghua Jiang, and Kui Ren. Toward privacy-preserving personalized recommendation services. *Engineering*, 4(1):21–28, 2018.

[49] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244. ACM, 2015.

[50] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. Blurme: Inferring and obfuscating user gender based on ratings. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 195–202. ACM, 2012.

[51] Eric W Weisstein. Bonferroni correction. 2004.

[52] Gregory L Wittel and Shyhtsun Felix Wu. On attacking statistical spam filters. In *CEAS*, 2004.

Appendix

A: Failure Analysis of the Gender Inference Model

The used gender classifier fails to predict the gender of some users in the dataset (see section 3.2). It is interesting to see which users cause difficulties to the classifier. This section investigates three different groups of users: users with a *lot of ratings*, users with *few ratings*, and *loyal users*. One might assume that users with only few ratings are more difficult to classify, because they do not provide enough data. On the opposite side, users who rated a large number of movies might be extremely easy to classify. Lastly, users that are loyal to a certain genre, should also be easily classified, except when they are loyal to a genre that is non-typical for the users' gender.

To test the performance of the classifier on these special users, it is trained on the training set and tested on the testing set, but all irrelevant users are filtered out. That means, in the case of *few ratings*, the testing set contains only users that rated less than a certain number of movies. While the attributes *lot of ratings* and *few ratings* are self-explanatory, we need to specify how the loyalty of a user to a certain genre is calculated. In principle, a user is loyal to a genre if (s)he rated mostly movies of one genre. If a user rated all genres equally often, (s)he is considered diverse. Thus, the largest percentage of a rated genre is suitable for a loyalty score. This percentage is calculated by dividing the number of ratings for the most rated genre by the total number of ratings per genre of that user. To give an example, imagine a user rated romantic movies 10 times, horror movies 3 times and sci-fi movies 5 times. That specific user would have a loyalty score of $\frac{10}{10+3+5} \approx 0.56$. However, there are some genres that frequently co-occur with another genre, like Drama and Romance. This makes a loyalty score of above 0.5 for the Romance genre difficult, because every time a Romance movie is rated, a Drama movie is rated as well, because there are numerous movies that are both Drama and Romance. Figure 9 shows the co-occurrence matrix of each genre in the MovieLens 1 million dataset. One can see that the co-occurrence of Romance and Drama, Drama and Comedy and Comedy and Romance is immense (the color is very bright in the heat map). Hence we combine these pairs and consider the pair Romance&Drama as Drama, Romance&Comedy as Comedy and Comedy&Drama as Drama.

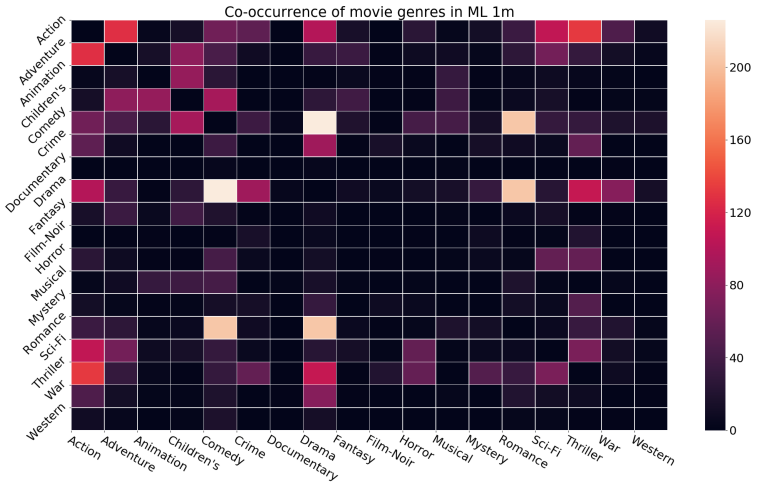

Figure 9: The co-occurrence matrix of genres in the MovieLens 1 million dataset as heat map. Bright colors indicate high values, darker colors indicate small values.

Table 27 shows the performance of the gender classifier on the specific users. The boundaries are chosen by hand, so that each testing set has about 100 users. One can see that the classifier performs better on the users with a lot of ratings than the users with only a few ratings. The findings also show that the classifier does not need a lot of ratings per user to classify reliably (AUC=0.80 for users with less than 26 ratings). This is in line with our previous findings in section 4.1.

Interestingly, the classifier seems to have difficulties with very loyal users. To understand why this is the case, we need to take a look at these users. Naturally, we cannot look at all 97 users from the set with boundaries 0.42 and 1.00, but we can look at a subset of them. Table 28 shows 7 randomly selected users with a loyalty score of 0.42 or more and their respective ordering of favorite movie genres. One can see that all males have Drama as favorite genre, which is interesting, because in section 4.4, we found that a preference for drama movies is indicative for a female user. It is most likely the case that the classifier has problems classifying these users.

	boundaries	AUC	N
all users	—	0.86	1208
very few ratings	20-26	0.80	96
few ratings	27-33	0.75	96
high nr. of ratings	282-389	0.88	98
very high nr. of ratings	390-1000	0.91	96
very low loyalty	0.0-0.17	0.78	102
low loyalty	0.17-0.19	0.85	102
high loyalty	0.35-0.42	0.79	103
very high loyalty	0.42-1	0.72	97

Table 27: Performance of the classifier on users with specific attributes. The column "N" contains the number of samples in the test set.

User	Gender	1st favorite genre	2nd favorite genre	3rd favorite genre
1	M	Drama (123)	War (15)	Action (10)
2	M	Drama (148)	Thriller (14)	Mystery (7)
3	F	Drama (80)	Thriller (9)	Crime (7)
4	F	Drama (229)	Thriller (28)	Action (18)
5	F	Drama (231)	War (18)	Thriller (12)
6	F	Drama (122)	Action (10)	Thriller (8)
7	M	Drama (98)	War (9)	Action (6)

Table 28: 7 randomly selected users with a loyalty score of 0.42 or more and their respective ordering of favorite movie genres. The number in brackets are the number of ratings for that specific genre.

The failure analysis reveals that gender inference might be more difficult for users that are loyal to certain genres. For the obfuscation, one could exploit that and make diverse users more loyal to a certain genre. Also, the gender of users that rated a high amount of movies are very easy to classify (AUC=0.91). That implies that obfuscation techniques should remove ratings from those users.

B: The Paper submitted to RecSys 2019 Workshop

We (Martha Larson, Manel Slokom and Christopher Strucks) submitted a paper about the BlurM(or)e algorithm (i.e. section 5) to the workshop of the RecSys conference 2019 and it got accepted. This paper can be found on the next few pages.

BlurM(or)e: Revisiting Gender Obfuscation in the User-Item Matrix*

Christopher Strucks
Radboud University
Netherlands
chr.strucks@gmail.com

Manel Slokom
TU Delft
Netherlands
m.slokom@tudelft.nl

Martha Larson
Radboud University and TU Delft
Netherlands
m.larson@cs.ru.nl

ABSTRACT

Past research has demonstrated that removing implicit gender information from the user-item matrix does not result in substantial performance losses. Such results point towards promising solutions for protecting users' privacy without compromising prediction performance, which are of particular interest in multistakeholder environments. Here, we investigate BlurMe, a gender obfuscation technique that has been shown to block classifiers from inferring binary gender from users' profiles. We first point out a serious shortcoming of BlurMe: Simple data visualizations can reveal that BlurMe has been applied to a data set, including which items have been impacted. We then propose an extension to BlurMe, called BlurM(or)e, that addresses this issue. We reproduce the original BlurMe experiments with the MovieLens data set, and point out the relative advantages of BlurM(or)e.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender Systems, Privacy, Data Obfuscation

1 INTRODUCTION

When users rate, or otherwise interact with items, they may be aware that they are providing a recommender system with preference information. Less likely, is, however, that users know that interaction information can implicitly hold sensitive personal information. In this paper, we focus on the problem of binary gender information in the user-item matrix, which can be inferred by using a gender classifier. The state of the art in gender obfuscation for recommender system data, is to our knowledge, represented by Weinsberg et. al. [11], who propose a gender obfuscation approach for a user-item matrix of movie ratings, called BlurMe. Successful obfuscation means that a user's gender cannot be correctly inferred by a classifier that has been previously trained on other users' rating data. BlurMe accomplishes this obfuscation without a substantial impact on the prediction performance of the recommender system that is trained on the obfuscated data. Our study of BlurMe has revealed that it has a serious shortcoming. In this paper, we discuss this issue, and propose an extension to BlurMe, called BlurM(or)e, that addresses it. We test BlurM(or)e against a reimplementation of BlurMe, reproducing experiments from [11].

Presented at the RMSE workshop held in conjunction with the 13th ACM Conference on Recommender Systems (RecSys), 2019, in Copenhagen, Denmark.

Obfuscation is an important tool to maintaining user privacy, alongside other tools such as encryption. Obfuscation is widely studied in other areas, but does not receive a great amount of attention in the area of recommender systems, exceptions are [2, 8]. Obfuscation can be added to the user-item matrix by users themselves, freeing them from an absolute dependency on the service provider to secure their data and use it properly. In [1, 2] the user can decide what data to reveal and how much protection is put on the data. Even trusted service providers can have issues, such as breaches, or data being acquired and used inappropriately [7].

The main contributions of this paper are:

- A discussion of a flaw we discovered in BlurMe.
- An extension to BlurMe, called BlurM(or)e, that addresses this issue.
- A set of experiments, whose results demonstrate the ability of BlurM(or)e to obfuscate binary gender in the user-item matrix with minimal impact on recommendation performance.

The paper is organized as follows. In Section 2, we cover the related work, before going on to present the shortcoming of BlurMe and our proposed improvement BlurM(or)e in Section 3. Next, we present our experiments and results in Section 4, and in Section 5 we discuss our reproduction of BlurMe[1]. We finish in section 6 with a discussion and conclusion.

2 BACKGROUND AND RELATED WORK

In this section, we discuss work most closely related to our own.

2.1 Obfuscating the User-Item matrix

In order to protect user demographic information in the user-item matrix, researchers have suggested data obfuscation. Data obfuscation (a.k.a. data masking) describes the process of hiding the original, possibly sensitive data with modified or even fictional data [10]. The goal is to protect the privacy of users, while maintaining the utility of the data. Data obfuscation can be done in several ways, e.g., [9] used lexical substitution as obfuscation mechanism for text or [5] used user groups instead of individual users to hide personal information from the recommender system. In BlurMe [11], the authors found that it is possible to infer the gender of users from their rating histories via basic machine learning classifiers. They proposed an algorithm, BlurMe, which successfully obfuscates the gender of a user, thereby blocking gender inference. BlurMe basically adds ratings to every user profile that are typical for the opposite gender, and is currently state of the art. The best performing BlurMe obfuscation strategy, the *greedy* strategy, decreases the

[1]The code for the reproduction as well as for BlurM(or)e and the exploratory analysis that we carried out is available at https://github.com/STrucks/BlurMore

accuracy of a logistic regression inference model from 80.2% on the original data to 2.5% on the obfuscated data (adding 10% extra ratings). The other proposed strategies have a smaller impact on the classification accuracy. For this reason, in this work, we focus on, and extend, the greedy strategy. Details and more explanations about the gender inference and obfuscation process can be found in section 5.

2.2 Inference on the User-Item matrix

The goal of BlurMe obfuscation is to protect against gender inference. The BlurMe [11] authors use basic machine learning models that can successfully infer users' gender from the user-item matrix. The most recent work on inference on the user-item matrix is, to our knowledge, that of [6], who developed a deep retentive learning framework that beats the conventional, standard machine learning approaches in the task of inferring user demographic information. For gender inference, [6] achieves a classification accuracy of 82%. However, this is only 2% better than the standard logistic regression model used in [11]. We adopt the model from [11] here since it is sufficiently close to the state of the art for our purposes.

3 BUILDING A BETTER BLURME

3.1 The Issue with BlurMe

BlurMe [11] proposes a powerful algorithm that can obfuscate the gender of a user. However, BlurMe has an important flaw: If the rating frequency of the movies are visualized, it is possible to determine that BlurMe has been applied to the data set, and to identify the movies for which ratings have been added. In figure 1(A), the rating frequency is shown for 20 items from the MovieLens data set before obfuscation. In figure 1(B), the rating frequency is shown for the same 20 items after obfuscation with BlurMe. BlurMe exhibits sharp spikes of items; here, it is item ID 27 (called *Persuasion*), which is marked in red. These spikes indicate that BlurMe has been applied, and point to the movies for which ratings have been added. There are two dangers associated with these spikes. First, if BlurMe is running at an operating point of 10% extra ratings using the greedy strategy, as mentioned above, then the gender inference accuracy is 2.5%. This means that if the information is known that BlurMe has been applied, it is simple to reverse the decision of the classifier, and gender can be known with an accuracy of 97.5%. Second, if we do not know the operating point of BlurMe (<10% extra ratings will not guarantee us a gender classification accuracy that we can reverse), we still can find the spikes in the rating histogram, and attempt to reverse BlurMe. In order to find a BlurMe spike we would look for movies that are known not to be particularly popular, but still have a lot of ratings in the BlurMe data. In this paper, we focus on addressing the first danger, and leave the second to future work.

3.2 The Definition of BlurM(or)e

BlurM(or)e was inspired by an exploratory analysis that we carried out, which revealed that a large number of movies are indicative of a gender. For this reason, it is not necessary to restrict the algorithm to add ratings only to the most correlated movies (like the greedy strategy of BlurMe does). This means that we can mask the data without heavily relying on a small set of movies indicative of gender.

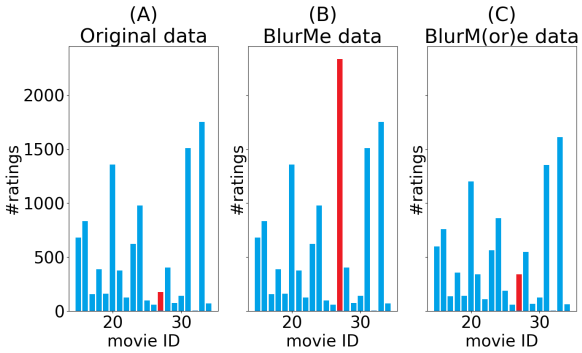

Figure 1: *#ratings* per movie for the movies 15 to 35. The red bar indicates an example of obvious data obfuscation after BlurMe is applied. The BlurMe data was created with the greedy strategy and with 10% extra ratings. The BlurM(or)e data contains also 10% extra ratings.

Based on these insights, we designed BlurM(or)e, which works as follows: We create, just like BlurMe, two lists of movies, L_f *and* L_m, that correlate most strongly with females and males respectively. After that, we alter every user profile by adding movies from the opposite gender list with the greedy strategy proposed in BlurMe [11]. However, if a movie has already doubled its initial rating count, it will be removed from the list. (We use $\times 2$, i.e., doubling, in this paper because it works well, and leave exploration of other possible values to future work). Also, we keep track of the number of added ratings, so that we can remove the same number later on. After every user has received extra ratings up to a fixed percentages of their original ratings, we remove ratings from users that have rated a lot of movies (here we choose $\geqslant$ 200 movies, although future work could investigate other values). The idea is that these users provide already enough data for the gender classifier, so removing some of their ratings would not impact the classifier. This idea is also inspired by our exploratory analysis, which revealed that the gender classifier does not benefit from additional data once a user has already provided 200 ratings. This removal would be more difficult to diagnose in the user-item matrix, since exact information of the rating rates about users would need to be available.

4 EXPERIMENTS AND MAIN RESULTS

4.1 Data

This study uses the publicly available MovieLens data set[2]. We chose MovieLens 1M, which is also used by BlurMe [11], whose work we are reproducing and extending. MovieLens 1M contains 3.7K movies and about 1M ratings of 6K different users, and also information on binary user gender. It is important to note that the distribution in the data set is unbalanced: there are 4331 males that produced 750K ratings and 1709 females that produced 250K ratings. Statistics of the original and the obfuscated data sets, are summarized in Table 1. We note that the number of items decreases for BlurM(or)e data sets due to the fact that the algorithm might remove all ratings of a certain movies by accident.

[2]https://grouplens.org/datasets/movielens/

Table 1: Statistics of the data sets used in our experiments and analysis.

data set	#Users	#Items	#Ratings	Range	Av.rating	Density(%)	Variance
MovieLens 1m	6040	3706	1.000.209	[1,5]	3.58	4.47	1.25
BlurMe 1% extra ratings	6040	3706	1.013.416	[1,5]	3.58	4.53	1.25
BlurMe 5% extra ratings	6040	3706	1.052.886	[1,5]	3.58	4.70	1.20
BlurMe 10% extra ratings	6040	3706	1.099.545	[1,5]	3.57	4.91	1.16
BlurM(or)e 1% extra ratings	6040	3705	1.000.797	[1,5]	3.57	4.47	1.24
BlurM(or)e 5% extra ratings	6040	3700	1.000.773	[1,5]	3.55	4.48	1.22
BlurM(or)e 10% extra ratings	6040	3699	1.000.395	[1,5]	3.57	4.48	1.16

4.2 Comparison of BlurMe and BlurM(or)e

We compare the performance of our new obfuscation mechanism, BlurM(or)e, with the original obfuscation mechanism BlurMe. The performance is measured, in line with the experiments in BlurMe, by the classification accuracy of a logistic regression model that is trained on unaltered data, and tested on obfuscated data. The performance is cross-validated using 10-fold cross-validation. Table 2 shows that BlurM(or)e performs similarly to BlurMe. The more obfuscation is applied to the data set, the lower the classification accuracy is. Note that Table 2 contains the reproduction of BlurMe that is discussed in detail in section 5. A big advantage of BlurM(or)e

		Extra ratings			
Classifier	**Data set**	0%	1%	5%	10%
Logistic Regression	BlurMe	0.76	0.54	0.15	0.02
Logistic Regression	BlurM(or)e	0.76	0.64	0.36	0.19
Random Classifier	Original	0.50	0.50	0.50	0.50

Table 2: Gender inference results measured in accuracy on BlurMe (reproduction) and BlurM(or)e

is that an attacker cannot easily see that the data set is obfuscated. Figure 1 on the previous page shows the number of ratings per movie for 20 different movies in the MovieLens 1m data set. The red bar corresponds to the number of ratings for the movie with ID 27. After the BlurMe obfuscation is applied, the red bar spans approximately ten times its original size. This makes the attacker suspicious and indicates that the data set is obfuscated. However, if the BlurM(or)e obfuscation is applied, the red bar only doubles its size, which is less noticeable. Also, BlurM(or)e has more similar statistics to the original data. Table 1 shows that BlurM(or)e keeps the number of interactions as well as the density similar to the original MovieLens data set, while BlurMe produces a more dense data set with more interactions.

The reduction part of BlurM(or)e has a less noticeable effect on the data set. Since the ratings are removed randomly from users with an extreme number of ratings, the number of ratings per movies distribution does not change dramatically (the bar with ID 20 shrinks $\approx$ 10% of its original size in the BlurM(or)e data set).

4.3 Recommendation Performance

Using a well known collaborative filtering technique, Matrix Factorization [4], similar to BlurMe, we notice that the change in RMSE is not substantial. The change has a maximum of 0.0298 for Movielens with BlurM(or)e and 0.0381 for BlurMe (with greedy strategy and 10% extra ratings). We can see in Table 3 that the RMSE is decreasing with an increase in obfuscation. BlurMe [11] discovered the same effect and explained that this might be due to the density of the obfuscated data. Since BlurM(or)e does not increase the overall density of the data, an alternative explanation can be found. The reason, lies perhaps, in increasing the density of users with few ratings.

	Extra ratings			
Obfuscation	0%	1%	5%	10%
Original	0.8766	—	—	—
BlurMe	0.8766	0.8686	0.8553	0.8385
BlurM(or)e	0.8766	0.8711	0.8640	0.8468

Table 3: The RMSE performance with Matrix Factorization on the original data, BlurMe data and on BlurM(or)e data.

5 BLURME REPRODUCTION IN DETAIL

Since we did not have the code of the original BlurMe [11], we reimplemented it in order to carry out the comparison in this paper. Because the paper was not specific about the settings of all parameters, it is not possible to create an exact replication. For completeness, we discuss our reimplementation here, so that authors building on our work have the complete details.

5.1 Gender Inference

This section describes our reimplementation of the gender inference models. We create the user-item matrix by associating every user with a vector of ratings: x_i with i being the index of the user and $x_{i,j}$ being the rating of user i for movie j. If the movie is not rated, we set $x_{i,j} = 0$. This results in a $U\,x\,I$ matrix, where U is the number of users and I is the number of items. Every user vector is associated with a gender, that will serve as target label for the classifier.

Following the experiments of [11], all classifiers are trained and tested on this user-item matrix with 10-fold cross-validation. We do not have information about the splits that were used, so we use our own splits. The ROC area under the curve as well as precision and recall are reported as performance measures. A comparison of the results can be seen in Table 4. The SVM uses a linear kernel and a C value of 1. For the Bernoulli classifier, the user-item matrix is transformed, so that every rating $x_{i,j}$ that is greater than 0, is set to 1. This means that the Bernoulli Bayes classifier ignores the value of the rating and only uses information about whether a user i rated the movie j or not. All remaining parameters for the other classifiers are set to the default values.

There is about a 4% difference between the scores reported in the original BlurMe paper [11], and those we measured with our

reproduction. Further exploration revealed that normalization, in terms of scaling all ratings from values in $[0, 5]$ to values in $[0, 1]$, can have a large impact on scores. We do not focus on normalization further here, but point out its impact because it suggests that there are parameters that could have been adjusted that are not explicitly recorded in [11]. In this paper, we have chosen to focus on the logistic regression model, since it is the fastest and achieves the best performance.

	BlurMe results		**Reproduction Results**	
Classifier	AUC	P/R	AUC	P/R
Bernoulli	0.81	0.79/0.76	0.77	0.88/0.48
Multinomial	0.84	0.80/0.76	0.81	0.89/0.77
SVM	0.86	0.78/0.77	0.79	0.83/0.82
Logistic Regression	0.85	0.80/0.80	0.81	0.84/0.83

Table 4: Gender inference results for both, BlurMe and the reproduction thereof. The performance is measured in ROC AUC, precision and recall.

Note that Table 4 uses ROC AUC as performance metric and Table 2 uses classification accuracy. This choice was made by BlurMe and for the sake of comparing the models, we did the same.

5.2 Gender Obfuscation

This section describes our reimplementation of the obfuscation approach of BlurMe [11]. Recall that the basic idea of BlurMe is to add fictional ratings to every user that are atypical for the their gender. BlurMe [11] creates two lists, L_f and L_m, of atypical movies for each gender by training and cross-validating a logistic regression model on the training set. The movies in L_f and L_m are ranked according to their average rank across the folds. The rank of a movie within a fold is determined by its coefficient that is learned by the logistic regression model. The lists L_f and L_m also include the average coefficient over all folds for each movie that serve as correlation metric between the movie and the user's gender.

After these lists are created, BlurMe takes every user profile and adds k fictive ratings to the profile for movies from the opposite gender list. The parameter k limits the number of extra ratings and is set to 1%, 5% or 10% in the original experiments. A male user with 100 ratings in the original data set would be obfuscated by adding 5 (for $k = 5\%$) fictive ratings from the female list.

There are some design choices left: Which movies should be selected from the lists and what should the fictive rating be? The authors of BlurMe [11] proposed three different selection strategies for the first problem: the *Random Strategy*, the *Sampled Strategy* and the *Greedy Strategy*. The *Random Strategy* chooses k movies uniformly at random from the list, the *Sampled Strategy* chooses k movies randomly, but in line with the score distribution of the movies. Thus, a movie that has a high coefficient is more likely to be added. Finally, the *Greedy Strategy* chooses the movie with the highest score. The authors do not mention the length of the lists, thus we chose to include all movies with a positive coefficient in the L_f list, and all movies with a negative coefficient in the L_m list.

For the fictive rating of a user A for a movie B, BlurMe suggests using either the average rating for movie B or the predicted rating for user A for movie B. Since [11] reports that there is almost no difference between these approaches, we chose to set the extra ratings for a movie according to its respective overall average rating. This average is rounded, because only integer ratings are valid.

The authors of BlurMe take the following attack protocol into account: A gender inference model is trained on real, non-obfuscated data and tested on the obfuscated data. For this reason, the gender inference model is trained on unaltered data and tested on obfuscated data. They use 10-fold cross-validation and report the average classification accuracy of the model.

We report results achieved by our BlurMe reproduction in Table 5. The reproduction is generally congruent with the original. The difference is negligible, we can see that the classification accuracy decreases if the obfuscation increases.

		Extra ratings			
	Strategy	0%	1%	5%	10%
BlurMe	*Random*	0.802	0.776	0.715	0.611
	Sampled	0.802	0.752	0.586	0.355
	Greedy	0.802	0.577	0.173	0.025
Reproduction	*Random*	0.76	0.74	0.69	0.62
	Sampled	0.76	0.71	0.58	0.33
	Greedy	0.76	0.54	0.15	0.02
Reproduction, Normalized	*Random*	0.81	0.80	0.78	0.76
	Sampled	0.81	0.80	0.78	0.75
	Greedy	0.81	0.78	0.74	0.70

Table 5: Performance of BlurMe's and the reproduction's obfuscation algorithm measured by classification accuracy.

6 DISCUSSION & CONCLUSION

In conclusion, this work points to a weakness in a state-of-the-art gender obfuscation algorithm, BlurMe [11], and presents an improved algorithm, BlurM(or)e, that addresses the issue. BlurM(or)e is shown to be able to obfuscate gender in the user-item matrix without substantial increase in RMSE. In other words, it keeps the utility of the data set intact. This work has shed light on some of the challenges of gender obfuscation.

We finish with a discussion of points from [11] that should be taken into account in future research. As mentioned before, normalization of the data set can have an enormous impact on the classification performance. In Table 5, we see that when our reproduction incorporates normalization the accuracy of gender inference still decreases with increasing obfuscation, but at a much slower rate.

In addition, BlurMe used the ROC area under the curve metric for the first gender inference experiments, yet changed to classification accuracy for the gender inference on the obfuscated data set. Using accuracy as a performance metric on imbalanced data sets is a practice that should be avoided. It is advised to report the ROC AUC, precision-recall AUC and ROC AUC on skew-normalized data when dealing with imbalanced data sets [3].

Finally, BlurMe declares (in [11]) the classification accuracy of 2.5% as a success. One can argue that the gender is only truly obfuscated if an attacking model achieves the same performance as a random classifier (i.e., exactly 50% accuracy, in the case of binary classification). This point should be taken into account in deciding the operational settings for BlurMe or BlurM(or)e. The decision also needs to consider the ease with which it is possible to detect whether a user's data has been obfuscated. Future work will study possibilities for obfuscating obfuscation.

REFERENCES

[1] Shlomo Berkovsky, Yaniv Eytani, Tsvi Kuflik, and Francesco Ricci. 2007. Enhancing Privacy and Preserving Accuracy of a Distributed Collaborative Filtering. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07)*. ACM, 9–16.

[2] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. 2012. The Impact of Data Obfuscation on the Accuracy of Collaborative Filtering. *Expert Systems with Applications* 39, 5 (2012), 5033–5042.

[3] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. 2013. Facing Imbalanced Data—Recommendations for the Use of Performance Metrics. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*. IEEE, 245–251.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer Society Press* 42, 8 (2009), 30–37.

[5] Dongsheng Li, Qin Lv, Li Shang, and Ning Gu. 2017. Efficient Privacy-Preserving Content Recommendation for Online Social Communities. *Neurocomputing* 219 (2017), 440–454.

[6] Yongsheng Liu, Hong Qu, Wenyu Chen, and SM Hasan Mahmud. 2019. An Efficient Deep Learning Model to Infer User Demographic Information From Ratings. *IEEE Access* 7 (2019), 53125–53135.

[7] Roger McNamee and Sandy Parakilas. 2018. The Facebook breach makes it clear: data must be regulated, The Guardian. https://www.theguardian.com/commentisfree/2018/mar/19/facebook-data-cambridge-analytica-privacy-breach, Online; accessed 05-July-2019.

[8] Rupa Parameswara and Douglas M. Blough. 2007. Privacy Preserving Collaborative Filtering Using Data Obfuscation. In *2007 IEEE International Conference on Granular Computing (GRC '07)*. IEEE, 380–380.

[9] Sravana Reddy and Kevin Knight. 2016. Obfuscating Gender in Social Media Writing. In *Proceedings of the 2016 EMNLP Workshop on NLP and Computational Social Science*. ACL, 17–26.

[10] Vicenç Torra. 2017. *Data Privacy: Foundations, New Developments and the Big Data Challenge*. Springer International Publishing, Cham, 191–238.

[11] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. 2012. BlurMe: Inferring and Obfuscating User Gender Based on Ratings. In *Proceedings of the 2012 ACM Conference on Recommender Systems (RecSys '12)*. ACM, 195–202.