Privacy-preserving Cross-Domain Recommender System for Healthcare Applications

Aslı Karahan

TU Delft

Privacy-preserving Cross-Domain Recommender System for Healthcare Applications

by

Aslı Karahan

to obtain the degree of Master of Science at the Delft University of Technology,
to be defended publicly
on Thursday, 7 July 2022, at 14:00

Thesis Committee:	Dr. Zekeriya Erkin	TU Delft, supervisor
	Dr. Burcu Külahçıoğlu Özkan	TU Delft
	Dr. Florian Hahn	UT Twente
	Dr. Nadia Metoui	TU Delft
	Jelle Vos	TU Delft
Institution:	Faculty of EEMCS, TU Delft	
Project Duration:	8 November 2021 - 7 July 2022	

Cover Image: Bosphorus view from my patient room in GATA Haydarpaşa Training Hospital
by Aslı Karahan, 12 May 2020

TU Delft

Preface

As one chapter of my life is closing and I step out into the professional world, I present to you, dear reader, my thesis. Herein, you will find the end result of eight whole months of hard work and dedication. Starting at the beginning of November, I spent the first three months absorbing information and searching for a research gap. When I finally found one, I was very excited to tackle this problem and come up with a solution of my own. The rest of my thesis process has been focused on developing a design to address the privacy and performance problems I identified, implementing, and evaluating. Now at the final stage, I am proud to present to you my work in its entirety.

My journey has been a positively challenging one that I could not have done alone. I want to wholeheartedly thank my supervisor, Dr. Zeki Erkin, for all the guidance, support, and motivation he provided me both professionally and personally. Aside from teaching me how to do research, he inspired me to always challenge myself, ask questions, and be confident in my endeavors. I also want to thank my daily supervisor Jelle Vos for all of the weekly brainstorming sessions and valuable feedback that shaped my thesis to its current form, Tianyu Li for all of his help in the differential privacy topic, and all the other cybersecurity Ph.D. students for always being open to talking. As my thesis progressed into other research areas, I was lucky to have discussions with researchers in the Multimedia Computing Group and the Computer Graphics Group. They were very welcoming to my endless questions in their respective areas. I want to also thank Dr. Burcu Külahçıoğlu Özkan, Dr. Florian Hahn, and Dr. Nadia Metoui for being part of my graduation committee and devoting their valuable time to evaluating my work.

Finally, I am blessed to have an extended support network of people I care for. I want to thank my family and friends for their endless support in studying at TU Delft and beyond: Thank you, Egon, for all the after-lunch cappuccinos and Italian lessons, Armin, Linus, Sven for all the pep talks, Roland, Li and Sandra for the lovely lunch breaks, Tarik for being an on-demand cat content provider and many others that I am yet to name. Charlie, thank you for being my best friend and the best course partner anyone can ask for. I want to thank my aunts, my uncles, my cousin, my brother, and his wife for being only a simple call away from me while being three thousand kilometers apart. Finally, many thanks to my mom, whose unconditional love and support allowed me to achieve my dreams.

Aslı Karahan
Ankara, June 2022

Abstract

Healthcare recommender systems emerged to help patients make better decisions for their health, leveraging the vast amount of data and patient experience. One type of this system focuses on recommending the most appropriate physician based on previous patient feedback in the form of ratings. Such advice can be challenging to generate for new patients as their interaction with the new environment is too limited to inform meaningful recommendations. Their preferences from the previous system can be transferred to the new system to help generate recommendations in a cross-domain approach.

Cross-domain recommender systems aim to bridge the gap between domains where such interactions are plentiful and domains with relatively little interactions for the new users, using transfer learning techniques. However, while privacy concerns in single-domain recommender systems have been studied extensively, the same does not apply to cross-domain systems where multiple domains collaborate. Privacy is essential in settings such as healthcare, where rating information is sensitive and prone to leaking patient medical history to third parties. Existing works lack an efficient design that provides sufficient privacy while generating recommendations.

This thesis presents a privacy-preserving cross-domain recommender system with partial user overlap. We first propose a privacy-preserving matrix factorization approach with differential privacy and multiparty setting to serve as the base of our cross-domain design. Our matrix factorization design with privacy is easy to adopt by existing systems, incurring negligible overhead, and requires no modification to existing factorization implementations. Next, we show the privacy-accuracy trade-off in our base method with experiments over the MovieLens dataset. Finally, in the cross-domain setup, we demonstrate that adding privacy to the proposed scheme has a minimal predictive performance loss, essentially providing privacy for free.

Contents

List of Figures

List of Tables

1 Introduction

More people move around the globe for work and study opportunities, family, and other reasons. Moving to a new location can be challenging in terms of settling down and finding a routine. It is ever more challenging to find the most appropriate healthcare professional for a patient profile in a brand new city, especially in the case of specific health conditions. While the patients can move around freely, their previous healthcare experiences, unfortunately, stay behind. People might ask around or use their social network to find physician advice. However, finding others with similar health experiences can be challenging while keeping a sense of privacy. It might also be unavailable in certain situations where patients' experiences and requirements are unique in their social environment, or they might find themselves in a new setting where such advice is unavailable [15].

Patient-centered decision-making processes correlate with improved health consequences [34]. This correlation can be due to the time, effort, and money spent searching for the most suitable physician and how the process affects the patient. The resources and especially time a patient spends searching for the best care can be spared by providing suitable recommendations. However, designing a system to generate appropriate recommendations is not trivial.

Leveraging the sizeable amount of health data outputted by many e-health applications, healthcare recommender systems (HRS) are developed to give relevant suggestions to patients [31]. The recommender systems can use patient ratings and reviews, along with the health and demographic profiles, to make recommendations. These systems differ in their approach to generating recommendations: from representing patients as nodes on a graph of their social interactions [38] to using distributed solutions such as blockchain to maintain groups of users [5]. Most solutions aim to find similar users or group them based on their similarities to generate meaningful recommendations. This approach is known as collaborative filtering. It has the underlying assumption that users with similar behaviors and ratings in the past will have similar tastes in future items.

Unfortunately, transferring the patient experience between different locations or domains is still not a common practice. For example, a patient might have plenty of experiences and data related to them in one location, as well as patient communities and similar profiles. However, that information does not transfer once the patient is in a different domain. In this thesis, we aim to address the knowledge transfer problem with a collaborative filtering approach emphasizing patient privacy. As it is crucial to improve the patient experience across domains, we believe the privacy of such a system should not be an afterthought.

1.1. Physician Ratings

For the physician recommender systems, ratings and general user feedback are essential to guide patients in searching for a healthcare provider. Online appointment platforms and web applications also have emerged to aid patients in their important healthcare decision, such as the DocPlanner Group. They are operational in 13 countries with different localized brand names. They serve over 80,000,000 patients looking for a physician each month as their self-reported statistics[8] reflect. With all of the information they collect, like personalized reviews, ratings, and rankings of physicians, it is important to show how in-demand physician recommenders are.

Another such example is WeDoctor, operating in China as an online appointment service provider [13]. They allow patients to search for doctors based on illness, category, proximity, and scores given by previous patients. They also host a rating and review system to help inform prospective patients of the previous experiences of others. Other services do not offer options to make an appointment but only host ratings and reviews on their platform, such as RateMDs. RateMDs is a platform operating under VerticalScope Inc. out of Toronto. They reportedly host 2 million doctor reviews and ratings that accumulated since 2004 [30].

With all information available online for patients to look for the best option for their health, there is also the problem of different regions and hospitals. Patients can see the ratings of doctors all over the world, but it does not necessarily translate into a localized and personalized recommendation. These services leave the decision-making to patients while only informing of subjective experiences of their users. When incorporated into a personalized recommender system for the users, the available ratings have several issues that need extensive consideration.

1.2. Cross Domain Recommendations

From a technical perspective, the online ratings services host ratings from *different system domains* as defined by [4]. This domain division is also present in the systems categorizing the physicians by the employed hospitals. From a recommender system perspective, combining all the ratings into one big rating matrix and generating recommendations is not practical. Due to different geographical boundaries and hospital locations, there would be big areas of non-existing ratings for each patient-physician pairing. Patients can not interact with physicians from many other cities or countries in a given location. The resulting rating matrix will be too sparse to generate accurate recommendations if the system domains are not separated.

Cross-domain recommender systems aim to overcome data sparsity and cold start problems, especially in collaborative filtering-based approaches. Collaborative filtering requires identifying similarly behaving users to generate recommendations. Data sparsity in CF settings results in overfitting and low accuracy [41]. Cross-domain approaches differ based on the assumptions of the domain compositions: item-user overlap scenarios, the direction of recommendation task, type of available domain data, and the number of domains [19]. Transfer learning techniques are developed depending on the scenario for which the recommendation task is to be improved.

Separating the physician ratings on a system level necessitates a cross-domain approach to making recommendations to avoid losing auxiliary information. Designing a cross-domain system in this setting also solves the new user problem in cases where the cold start user is already active in other domains. The cross-domain approach also addresses the issue of a patient moving to a new location and finding a physician that will be most relevant in the context of their previous experience. Transferring a patient's preferences to a different hospital can benefit their health condition as less time and resources would be spent by the patient until they can find the most appropriate physician.

1.3. Privacy Considerations

While providing meaningful recommendations that aid people in their health decisions is essential, the privacy aspect of recommendation generation must be considered. Physician recommender systems have stricter privacy requirements compared to other privacy-preserving recommenders. Even the information of a rating existing between a physician-patient pair is leaking personal information [15]. The existence of a rating, combined with publicly available information on the specialization of the physicians, can be combined to deduce the general ailment of a given patient. This information is highly personal, as it might also lead to discrimination of the individual based on a specific health condition, such as mental illness or pregnancy. An extensive health profile can be constructed for a patient given their ratings, which can be used for discriminatory purposes. This information used to generate recommendations can be exploited further to harm the patient by adjusting the insurance tariffs unfairly [32].

Not only the existence of ratings but also the rating values must be kept confidential. Keeping the ratings open to all public creates legal issues such as libel cases, intimidation, and reliability. RateMDs website has an entire section of legal and privacy-related questions, as lawsuits and problems are prevalent in their open system. In addition, rating healthcare experience openly might cause intimidation for patients. Patients might be reluctant to rate their experience due to worries about retaliation or losing the quality of the help they receive.

As shown by [32], patients are more open to sharing health data in scientific settings compared to commercial settings. They also discovered that patients are reluctant to share mental health-related data compared to other types of health data in healthcare recommender system settings. This perception of privacy and its utility is a trade-off in healthcare recommender systems and informs how to construct a better physician rating scheme. Patients would be more reluctant to share their ratings in a commercial open system that does not offer privacy features than one that offers proper protection against misuse.

Another problem with the open rating system is that the ratings are not open to users in a contextual setting. Therefore, when users see a review, they do not know how they are similar or dissimilar to the rater. This situation might result in overbooking more popular physicians who are not necessarily the best fit for a given patient.

Moreover, the collaboration aspect of a cross-domain setting also raises privacy issues. Institutions are often unable or reluctant to share private data due to regulations. Users might be reluctant to share their data with third parties for recommendations generated for other users. Without the ratings transferring among domains, the overlapping and non-overlapping user identifiers might be shared to establish a model. In addition, the domains might want to keep certain physicians' performance indicators private for competitive reasons.

Anonymization does not address the issues mentioned above. In our literature review phase, we have not come across any work on developing a privacy-preserving cross-domain recommender system with partial user overlap. The privacy-preserving recommender systems do not consider the cross-domain or new user scenarios. State-of-the-art cross-domain recommender systems do not address privacy issues. Surveys have noted this gap in research. In [41], developing privacy-enhancing solutions in cross-domain recommendation systems is the third most pressing issue, and [19] denoted privacy as an open research area.

1.4. Research Question

Based on the research gap in this field, we have developed the following research question:

How can we design an efficient privacy-preserving cross-domain recommender system with partially overlapping users?

This thesis aims to design a privacy-preserving cross-domain recommender system for new users and determine the proposed design's privacy accuracy and performance trade-offs. We use a publicly available rating dataset to evaluate our system and compare it with baseline implementations. We determine the cost of adding privacy to such systems in predictive performance and runtime.

1.5. Our Contribution

We present a novel design to perform privacy-preserving matrix factorization based on differential privacy and multi-party computation. Our design has two approaches to increase predictive performance while giving formal privacy guarantees. We also present a proof of concept implementation on the MovieLens 100k dataset for our design to demonstrate the accuracy and privacy trade-off of our privacy-preserving matrix factorization scheme.

We design a privacy-preserving cross-domain recommendation scheme using the best-performing privacy-preserving matrix factorization method. Our cross-domain recommendation task is specifically for the new, cold-start users with no available ratings in the target domain. We design the underlying cross-domain recommender using inter-domain similarities and intra-domain similarities. Finally, we implement a proof of concept to demonstrate the performance of the underlying recommender system and the effect of adding privacy.

We contribute an easy-to-adopt design where current recommender systems can be converted accordingly without high runtime costs or accuracy loss. Our privacy-preserving cross-domain recommender system work is the first in the field that provides rating privacy against the recommendation system servers and new user recommendations with accuracy evaluation.

1.6. Overview

Chapter 2 gives an overview of the techniques used in the design we present. Chapter 3 discusses the related literature and the current state of the art. Chapter 4 discuss the design, implementation, and evaluation of the privacy-preserving matrix factorization methods. We present our privacy-preserving cross-domain system's design, implementation, and evaluation details in 5. Finally, in Chapter 6 we conclude our thesis and discuss future work.

2 Preliminaries

This chapter discusses the preliminaries required to understand the rest of this thesis. We overview techniques and subjects such as matrix factorization, differential privacy, keyed hash functions, secure dot product, and error metrics.

2.1. Matrix Factorization

Collaborative filtering is a recommender system technique where recommendations are generated for a given user based on other users that are similar or dissimilar. The main idea behind this approach is the assumption that similarly behaving users in a system will continue to behave similarly in future interactions. However, as the rating matrices are generally highly sparse, with many item-user pairs missing a rating, it is challenging to implement a collaborative filtering approach. Therefore, techniques such as matrix factorization map the interaction space into a denser dimension to make collaborative decisions. The technique is most know from Simon Funk's blog post[12] where he details the technique that won him third place in Netflix Prize.

As an umbrella term, the matrix factorization problem in recommender systems refers to the decomposition of a rating matrix into two lower rank matrices, the multiplication of which results in the original rating matrix. Figure 2.1 gives an example representation of a factorization. This decomposition aims to discover underlying ”latent” factors of users and items to generate recommendations[21]. The inner product of a user's latent factor and the item's latent factor results in the approximation of the submitted rating or the prediction in the case of a missing rating.

Factorizing a given matrix is an optimization problem in which we approximate the two latent factor matrices. The error between the known rating values and the predicted values is minimized. In equation form, we can see it as:

$$\min \sum_{u,i \in R} \left(r_{ui} - \mathbf{U}_u^{\mathsf{T}} \cdot \mathbf{V}_i \right)^2 , \tag{2.1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are the user and item latent factors respectively. However, the optimization problem in this form is prone to overfitting. Therefore we include regularization terms as follows:

$$\min \sum_{u,i \in R} \left(r_{ui} - \mathbf{U}_u^{\mathsf{T}} \cdot \mathbf{V}_i \right)^2 + \lambda \left(||\mathbf{U}_u||^2 + ||\mathbf{V}_i||^2 \right) . \tag{2.2}$$

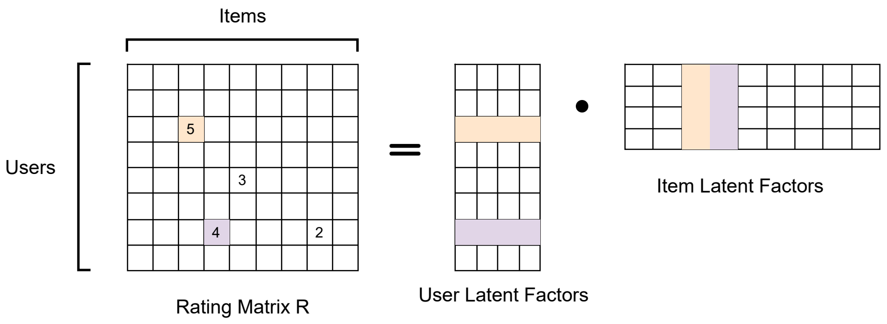


Figure 2.1: Visualising Matrix Factorization

The λ parameter determines the degree of regularization.

The assumption that we can capture all of the effects that lead to a rating through two vectors is strong. Other factors might lead to a user u giving an item i a particular rating. For example, some users might be more generous with their ratings, or some items might receive high ratings from many users. These signals are available in a rating matrix, as the recommender system can distinguish users with high and low average ratings as well as high and low rated items. Using the factorization to generate two matrices that explain all the variations in ratings is unrealistic because it distills the recommendation problem into modeling pairwise interactions only and disregarding available signals. Instead, recommender systems have evolved to include user-specific and item-specific biases contributing to a rating. In this notation, the rating of a user u regarding an item i is as follows:

$$r_{ui} = \mu + b_u + b_i + (\mathbf{U}_u)^{\mathsf{T}} \cdot \mathbf{V}_i, \tag{2.3}$$

where μ represents the global average, b_u and b_i are the user and item biases, and $(\mathbf{U}_u)^{\mathsf{T}} \cdot \mathbf{V}_i$ explains the interaction. Including the biases, the optimization problem transforms into

$$\min \sum_{u,i \in R} \left(r_{ui} - \mu - b_u - b_i - (\mathbf{U}_u)^{\mathsf{T}} \cdot \mathbf{V}_i\right)^2 + \lambda \left(||\mathbf{U}_u||^2 + ||\mathbf{V}_i||^2 + b_u^2 + b_i^2\right). \tag{2.4}$$

In our proof-of-concept, we use a library that implements the solution to this optimization problem with stochastic gradient descent(SGD). SGD works through iterations that modify the model values of $\mathbf{U}$, $\mathbf{V}$, b_u and b_i with a given step size and error value in each iteration.

2.2. Differential Privacy

Differential privacy is a privacy technique that allows databases to maintain functional statistical structure while preserving individuals' privacy. First formalized by Dwork et al. [10], it is a formal model of maintaining individual record privacy in a given dataset. The central intuition in our differential private recommender system setting is that two rating matrices that differ only by one rating should not reveal information about the differing entry based on computations they carry out in recommendation generation or the output they present to end-users.

We can construct a more straightforward example in a recommender system setting: our primary user requests a prediction on a given item. After observing another user participating in the recommendation scheme and submitting a rating, our primary user requests the prediction on the particular item again. The result will be different due to the new user contributing. Our primary user might differentiate the newly participating user based on the output change, the rating prediction. For example, suppose the predicted rating is lower than before. In that case, our primary user can deduce that the new user submitted a low rating on that item, hurting the new user's privacy in return.

In formal notation, we consider two datasets as D and D' from the set of all possible input spaces that are identical except for one element. We define A as a randomized algorithm and S as the possible output space of A. We say A is ϵ-differentially private if the following inequality holds:

$$Pr[A(D) \in S] \leq exp(\epsilon) \cdot Pr[A(D') \in S]\,. \tag{2.5}$$

The output distribution of the algorithm A should not change more than a factor of ϵ, as given in the inequality. As the ϵ increases, we can see that the output distribution changes more with respect to the differing element, which allows us to differentiate between the input dataset, thus providing less privacy but preserving more utility. Smaller ϵ means the output distribution is harder to distinguish as they differ by a factor of ϵ, thus higher privacy for the input sets.

We achieve this property by adding controlled noise to the data to simulate the effect of removing an element. In other words, the noise compensates for the "differing" element, so the output of the algorithm A does not leak information that would allow us to differentiate between D and D'. The noise is calibrated with respect to the sensitivity of the function Δf, the maximum amount an element can affect the distribution. In our recommender system setting, the ratings are in a given range. Hence the sensitivity is selected as the absolute difference between the maximum and the minimum possible rating value.

Dwork et al. [10] propose sampling noise from the Laplace distribution to achieve this property. The ϵ and the sensitivity variables are incorporated as follows:

$$Lap\left(mean = 0, scale = \frac{\Delta f}{\epsilon}\right). \tag{2.6}$$

2.3. Keyed Hash Function

A hash function is a function that can map an arbitrary input space to a fixed-length output space. Hash functions are one-way because of the size difference between the input and output spaces. A keyed hash function, more commonly known as hash message authentication code(HMAC), is a construction that involves a cryptographic hash function and a key. Cryptographic hash functions have properties that allow them to be functional for security purposes: preimage resistance, second preimage resistance, and collision resistance[29]. The first property addresses the one-wayness of the function as it should be infeasible to revert the hash operation and recover a meaningful message from the hash. The second preimage resistance property ensures that given a message, it should be infeasible to find another message that results in the same hash. Finally, the last property states that it should be infeasible to generate two different messages that result in the same hash output.

Cryptographic hash functions can be keyed by applying the secret key on the input of the hash function. To meet the three properties above, hash functions are designed such that changing the input changes the hash function's output significantly, also referred to as the "avalanche effect". Therefore, different keys on the same messages result in entirely different hash outputs. In addition, preimage resistance prevents the underlying message or the secret key from being leaked to an observer of the hash digest. Finally, keyed hash functions are also deterministic. Considering these properties, we use keyed hash functions to maintain item privacy in our design.

2.4. Secure Dot Product

We have two parties, Alice and Bob, each holding a vector of size n, A, and B. Using secure dot product operations, the parties aim to jointly calculate the dot product of the vectors without revealing the vectors to each other. In the protocol by Du and Zhan [9], Alice and Bob each end up with a share V of the resulting dot product as $V_A + V_B = A \cdot B$. For our construction, we want one of the parties to have the full dot product resulting from the computation. Therefore, we will be modifying the protocol. We assume an honest-but-curious model where the parties follow the protocol honestly, with correct inputs but try to learn from the information they can honestly gather.

For this construction, we also include a trusted third party to generate ”commodities” to Alice and Bob, an approach proposed by Beaver[2]. The commodity server provides security resources, in our case, a set of random vectors for Alice and Bob to conduct operations. The protocol is as follows:

1. The commodity server generates two random vectors of size n as R_a, R_b, and a random number as r_a.
2. The commodity server calculates r_b as $R_a \cdot R_b - r_a$.
3. The commodity server sends Alice and Bob commodities as: (R_a, r_a), (R_b, r_b).
4. Alice and Bob masks their vectors using their respective commodity vector and send each other the result. Bob receives $(\hat{A} = A + R_a)$ and Alice receives $(\hat{B} = B + R_b)$.
5. Upon receiving $\hat{A}$, Bob calculates $(\hat{A} \cdot B + r_b)$ and sends it to Alice.
6. Alice computes $(\hat{A} \cdot B + r_b) - (R_a \cdot \hat{B}) + r_a$. We see that this calculation simplifies to $A \cdot B$. Substituting $\hat{A}$ and $\hat{B}$ results with

$$\begin{aligned} &= ((A + R_a) \cdot B + r_b) - (R_a \cdot (B + R_b)) + r_a\,, \\ &= A \cdot B + R_a \cdot B + r_b - R_a \cdot B - R_a \cdot R_b + r_a\,. \end{aligned} \tag{2.7}$$

Simplifying $R_a \cdot B$ terms results with

$$= A \cdot B + r_b - R_a \cdot R_b + r_a\,. \tag{2.8}$$

Using the equality $R_a \cdot R_b = r_a + r_b$, as set up by the commodity server, we obtain

$$A \cdot B + r_b - (r_a + r_b) + r_a = A \cdot B\,. \tag{2.9}$$

The communication complexity is three times that of the regular dot product operation, excluding the overhead of the setup phase associated with the commodity server. Furthermore, due to the randomness of R_a, R_b and r_a, Alice and Bob are unable to learn about the private vector of the other from the information they receive throughout the protocol.

2.5. Error Metrics

Two primary metrics in evaluating recommender system performance are Root Mean Squared Error(RMSE) and Mean Absolute Error(MAE). These metrics aim to measure the predictive performance of recommender systems with rating prediction in the literature. In addition, their use allows for an interpretable performance discussion, as they both are in the rating range.

Mean Absolute Error is the average absolute error on all of the predictions. The formula is as follows:

$$MAE = \frac{1}{|R|} \sum_{u,i \in R} |r_{ui} - \hat{r}_{ui}| \, . \tag{2.10}$$

MAE gives equal weight to all errors, so the outliers and large errors will be weighted equally to other errors. Therefore, it gives a holistic interpretation of the recommender system's predictive performance and is widely common.

Root Mean Squared Error is the square root of the mean of squared error instances. The formula is as follows:

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{u,i \in R} (r_{ui} - \hat{r}_{ui})^2} \, . \tag{2.11}$$

As we take the square of the errors before averaging, the larger errors make the result bigger. So the larger error values affect the metric more than the smaller errors. In other words, RMSE penalizes large errors more severely compared to MAE.

3 Related Works

Recommender systems have been studied extensively over the last years to increase predictive performance, address different recommendation problems and add privacy components. This chapter discusses previous works that have laid the foundation for our work.

We group the literature into several sections demonstrating our work's components: In Section 3.1, we discuss the privacy-preserving medical recommendation systems. These works establish the privacy requirements of the physician recommendation systems and offer solutions in a single domain setting. Section 3.2 discusses privacy-preserving matrix factorization approaches as an essential part of our cross-domain construction. We present a summary of the papers from the first two sections in Table 3.1.

Following this, we discuss some papers from the cross-domain recommender systems without privacy in Section 3.3, mainly focusing on the ones with partial user overlap and solving the new user problem. The area of cross-domain recommender systems is vast; hence in this section, we only discuss the works most applicable to our physician rating system across different domains. Finally, we discuss the cross-domain recommender system constructions that are privacy-preserving in Section 3.4.

3.1. Privacy-preserving Physician Recommendation Systems

This section introduces previous work that designed privacy-preserving healthcare recommender systems. While it is a broad area of research, this section focuses on works that are the most similar to our problem and approach.

Katzenbeisser and Petkovic [17] is one of the earliest works in developing privacy-enhancing recommendations specific to the healthcare setting. They apply private profile matching techniques to tackle three problems in the healthcare setting: sharing the patient's health profile, physician-patient matching, and patient community creation. They use homomorphic public-key encryption schemes to allow vector operations by calculating the inner product, Euclidean distance, or subset matching over binary vectors. Finally, to vectorize the preferences or the patients' medical information, they propose a mapping phase where information is encoded and encrypted to be processed by a central server.

Their solution is a neighborhood-based approach based on the encoding of the medical data. Their proposed system takes auxiliary patient information and not submitted ratings to calculate similarities. In this sense, it is different from our work's rating-based problem. However, it is an important paper that sets the precedent of patients having control over their data and performing cryptographic operations locally.

Hoens et al. [15] propose two different designs for a physician recommendation system. Their *Secure Processing Architecture* design consists of patients submitting encrypted ratings related to a specific physician and the condition to one or multiple servers. The recommendation is generated based on the aggregation of the scores, and the highest-rated physician is recommended. They also address the possible reliability issues, such as submitting a rating outside of the available range by introducing Zero-Knowledge Proofs to prove the validity of the rating. In addition, their scheme considers professional experience. They add a variable value to the average rating of the physicians that have worked with more patients. While their design consists of several computational servers, recommendation generation is central. In addition, the recommendations are not personalized per user but only consider high average scores. Similar to the [17] work, they use an additively homomorphic encryption scheme to aggregate the submitted ratings.

Their second design is called *Anonymous Contributions Architecture* where the same rating aggregation approach is implemented without encrypted ratings but with anonymous users. In this design, reliability is not an issue of ratings being in an unauthorized range as the receiving server can easily discard it. However, they have tackled the issue with repeated submissions in an anonymous setting. While aggregating is straightforward as the ratings are in plaintext, they propose using anonymous credentials or e-cash based scheme to resolve the reliability issue.

As for their system's recommendation performance, they refrain from implementing predictive performance over a dataset. Instead, the authors mainly develop a secure rating aggregator with privacy and reliability per their focus.

Kaur et al. [18] is another paper in healthcare recommender systems that take a more current recommender system as the basis for their design and improve on it by adding privacy and a distributed environment. Their system is a single domain recommender system that assumes the rating data is arbitrarily distributed among several parties. They also base their design on an item-based collaborative filtering approach where they consider similarities between the items' ratings. They keep similarity calculation as an offline phase and only rating prediction as an online component to have an efficient design. For the similarity calculation, they use random masking for the dot product operation and additively homomorphic encryption for the magnitude calculation. One of their strongest assumptions is that the indexes of the items and users held by the parties are known. They test their design on a simulated dataset of physician ratings and the MovieLens dataset with a 70-30 train-test split and demonstrate no accuracy loss in their privacy-preserving scheme.

3.2. Privacy-preserving Matrix Factorization

This section presents research focusing on adding privacy components to matrix factorization processes for recommender system privacy and designing privacy-preserving single domain recommenders.

Nikolaenko et al. [24] is the first work in this field to allow for matrix factorization over the rating matrix to be carried out in an encrypted setting. They propose a parallelizable design using homomorphic encryption and Yao's garbled circuits. They do not specify any specific use-case for their system but perform experiments over MovieLens data and a synthetic dataset. They improve their implementation efficiency by parallelizing the sorting network and using a multi-threaded library. Despite their efforts, the execution time is high due to communication and computational costs. They report taking 2.9 hours for a rating set of 14.7k ratings on a machine with 16 cores. Although they argue their process can be made faster by running on more powerful hardware, their solution is impractical to be used in real-life systems due to performance issues.

Nayak et al. [23] further improved on the parallelism proposed on [24]. Their work introduces the GraphSC framework, where they provide a way for graph-based algorithms to be parallelized with a privacy-preserving data oblivious approach. Their proposed method applies to many big data tasks, including matrix factorization. They report being able to factorize the MovieLens dataset of 1 million ratings in 13 hours, which is way faster than the [24] attempt. While their method offers significant performance improvement in speed, the baseline of the current non-privacy preserving matrix factorization runtime is a matter of seconds for 20M ratings[1].

Kim et al. [20] has proposed an efficient design using fully homomorphic encryption and additively homomorphic encryption. They design a novel data structure that allows Single Instruction Multiple Data (SIMD) operations over the latent factor matrices, allowing a drastic runtime performance increase. They introduce a third party as Crypto Service Provider (CSP) in addition to the Recommender System (RecSys). RecSys and CSP communicate per each iteration of the stochastic gradient descent. Since the ratings and the latent factors are encrypted, they perform fixed-point arithmetic over the data. As such, they also evaluate the error with the size of the fractional part. To address the issue of hiding the existence of ratings, the information that a certain user rated a certain item, they propose adding fake ratings and flagging them to exclude in the factorization operations.

Despite having costly cryptographic primitives in their design, the runtime of their protocol is a significant improvement over the previous works. To compare with state-of-the-art, they perform matrix factorization over 14.7k ratings as [24] and report finishing in 1.5 minutes on a 6-core machine. Their runtime performance is 50 times better than the previous work while still being very slow compared to non-privacy preserving solutions.

While previous approaches perform operations over encrypted data, they are too impractical to adopt by the existing recommender systems due to high communication overhead and costly cryptographic techniques. This performance issue has led to the development of differentially private recommender systems. Various solutions propose adding differential privacy noise to different stages of the matrix factorization process to achieve privacy. According to [3], DP noise can be added to inputs (ratings), the gradients in the process of matrix factorization, and the outputs (user and item profiles). For physician rating systems, input privacy is essential, as demonstrated by the papers explained in Section 3.1. Therefore, this section also discusses differential privacy solutions that offer input perturbation.

Berlioz et al. [3] investigates the three ways of adding differential privacy to the matrix factorization operations and the performance implications of this perturbation. They also compare neighborhood-based approaches with the matrix factorization under the noise-added data. They conduct experiments over the three versions of the MovieLens dataset: 100k, 1M, and 10M variations. Their findings point to a privacy-accuracy trade-off; they report stronger privacy properties when the added noise value increases, but the model's predictive performance decreases. They also conclude that the best performing option is to add the noise to the input rather than adding it to gradients or the results as the noise accumulates when added in the learning process.

Zhaoyan et al. [16] explores the input perturbation approach further by designing a noise-adding process. Their noise generation consists of multiple layers and works by combining values from the uniform and Gaussian distributions. Their design ensures that the rating is still in the desired range before the user sends it to the recommendation server. Furthermore, their proposed layer system considers the maximum possible value the ratings can take. This design choice is not unlike differential privacy, where the Laplacian noise is generated considering the sensitivity.

[1]`https://github.com/gbolmier/funk-svd`

In addition to proposing a novel random perturbation method, they also design a rating filling system. In the proposed rating filling approach, the recommendation server splits the training data and first trains on the subset of the training data. The recommendation server then predicts unseen item-user pairs until the desired sparsity and then performs final training on the sparsity reduced data. They test their approach on three datasets: MovieLens 100k, FilmTrust, and MovieTweetings. With the variety of the datasets, they can argue robustness and test how the perturbation method performs with different rating ranges. They report similar findings to [3], especially the argument that there is a privacy and accuracy trade-off in the privacy-preserving matrix factorization methods. However, unlike other methods with differential privacy approaches, they cannot give a formal privacy guarantee to their proposed solution.

Table 3.1: Summary of Privacy-Preserving Physician Recommender Systems and Matrix Factorization Methods

Design	Privacy Guarantee	Item Obfuscation	Runtime Cost	Predictive Performance Loss
Katzenbeisser and Petkovic[17]	Homomorphic Encryption	Yes	High	Not applicable
Hoens et al. [15]	Homomorphic Encryption, ZKP, Anonymous Credentials, E-Cash	Yes	High	Not applicable
Kaur et al. [18]	Homomorphic Encryption, Random Masking	No	High	No
Nikolaenko et al. [24]	Yao's Garbled Circuits, Homomorphic Encryption, Oblivious Sorting	Yes	Highest	Low
Kim et al. [20]	Homomorphic Encryption	Yes	High	Low
Berlioz et al. [3]	Differential Privacy	No	Low	High
Zhaoyan et al. [16]	No formal privacy definition	No	Low	Low

3.3. Cross Domain Recommender Systems without Privacy

Designing cross-domain recommender systems is an active research area with different problem definitions and solution directions in the general recommender system ecosystem [4]. However, this section focuses only on the solely rating matrix factorization systems. Therefore, review-based and tag-based cross-domain approaches are out of the scope of this thesis.

Man et al. [22] propose an embedding and mapping approach improve cross-domain recommendation performance. Their design relies on learning the mapping between the latent factors across the domains. They propose to learn two different types of mapping: finding a transfer matrix to learn a linear mapping between the domains and a multi-layer perceptron for learning the non-linear mapping between the domains. They also include two different methods for latent factor modeling: matrix factorization and Bayesian Personalized Ranking. The domain mapping can be learned regarding the user or the item latent factors. Due to their datasets having overlapping items and no overlapping users, they opt to learn item-based latent factor mapping and generate recommendations accordingly. They experiment on different fractions of overlap between the domains, ranging from 10% to 50%, and show that user overlap percentage correlates with better performance for cold-start users. They demonstrate that they improve on the previous state-of-the-art [27, 1] predictive performance for both the linear and non-linear mapping approaches. Following their results, this work constitutes a benchmark against which predictive performance is measured against in the following works.

Wang et al. [33] propose a design specifically aimed at making predictions for the cold-start users across different domains. Their definition of different domains entails differences in item type, such as between movies, books, and electronic purchases. Similar to [22], they propose to learn the mapping between the latent factors of the source and the target domain and similarly propose a multi-layer perceptron for this task. However, they differ from previous work as they propose using gradient boosting trees. Additionally, they propose a variation on the matrix factorization process where they include user similarities from the target domain, resulting in the factorization of the auxiliary domain to capture target behavior better. Additionally, they adapt a neighborhood-based approach while considering the linked users across the domain to capture the most appropriate set of linked users to learn from while determining the mapping function. Their runtime incurs an overhead of similarity calculation, removing users and items associated with sparse ratings for better prediction generation. Overall, they demonstrate a significant predictive performance increase over [22]. Wang et al. also confirm that increased user overlap results in better predictions for cold-start users.

Sahu and Dwivedi [28] propose a different approach to the previous works. Previous works focused on domain adaptation methods: designing a system with a mapping-based idea where they solve the cross-domain problem through a mapping function or adaptation between the domains. Instead, Sahu and Dwivedi propose a system where they modify the objective function of the target domain with information from the source domain. The domain adaptation strategy they propose is present from the beginning of the matrix factorization. Their system penalizes the target domain factors as they differ from the source domain factors. The main idea behind this approach is that overlapping users should be represented ideally by the same factors in both domains as they belong to the same entity. Their approach is different from [27], as they do not aim to find a common coordinate space where entities are jointly present but project the target domain entities to their source domain representation as close as possible using a cost system. They report an improved predictive performance over the baselines they chose, but unfortunately, they do not compare with the state of the art [27, 1, 22]. They also do not comment on the usability, as their design allows for only one-sided knowledge transfer and requires repetition of the matrix factorization process each time a new transfer task is required. Their system also requires complete user overlap between the domains.

Some approaches also represent the domains as two separate, fully connected graphs and the overlapping users as bridges. The vertices represent the similarity between the users for the recommendation tasks for the non-overlapping entities. The similarities between not directly connected users are discovered and later incorporated into recommendation generation.

One such approach is proposed by **Xu et al.** [36]. Their design enhances an existing friend network using the similarity values calculated between the users. They achieve this enhanced transfer matrix by random walks over the transfer matrix. The similarity between the users is the cosine similarity over the submitted ratings. They base their cross-domain recommendations on a connected friend network of similar tastes. The similarities between the closest friends are normalized to add up to 1, and a weighted average of close friends is calculated with the normalized similarity values. Overall, they report an improvement over the error metrics compared to other neighborhood-based methods, but no comparison is present to previous cold-start user recommendation research.

Zhang et al. [39] also uses a similarity matrix between the users in the source and target domains. Their main problem is focused on cold-start users and on mutually improving the recommendations for overlapping and non-overlapping users. Due to this problem shift, they report better error values than other works focusing on cold-start issues, as they incorporate more information on the overlapping users. They calculate the similarities between users' latent factors after projecting onto a shared space. With this approach, they are now the current state of the art for cross-domain recommenders that aim to predict ratings without auxiliary information such as tags or reviews. Their work incorporates many of the approaches we discuss here, such as projecting on a common coordinate space[27], random walk over similarities[36]. Additionally, their approach is in line with other works that argue that calculating similarity over the latent factors results in a better understanding of the underlying similarity relation compared to the rating-based similarity methods[14]. Finally, they demonstrate that combining the latent factor approach with user similarity calculation improves cross-domain recommendation performance for partially overlapping users.

3.4. Privacy-preserving Cross-Domain Recommender Systems

As noted by multiple surveys on cross-domain recommendation systems [4, 41, 19], the privacy aspect of the cross-domain recommender systems is an issue that requires attention. They agree that the goal of transferring knowledge between domains might be a problem due to regulations around user privacy. Despite being an up-and-coming field, privacy-preserving cross-domain recommenders have little research to be discussed. This section discusses those which are rating matrix-based as it is the most relevant to our problem definition.

To the best of our knowledge, the first effort in designing a privacy-preserving cross-domain rating-based recommender system is by **Ogunseyi et al.** The authors made two publications on their design: [26] being a general framework of their approach and design specifics, [25] implementing their approach and giving a performance evaluation. The proposed design uses homomorphic encryption on user ratings to preserve user privacy. Due to homomorphic properties, the recommender system server can perform a stochastic gradient descent algorithm over the rating matrix. They introduce a privacy server to their construction that generates the key pair and distributes the public key to the end-users. The cross-domain aspect of their design is the concatenation of the latent factors of the source and the target domains to enrich the target domain model.

As they describe it, the authors' approach is problematic since the result of the concatenation operation is not usable later in the dot product with the item latent factors due to dimension mismatch. They do not address this issue in their design at [26] as they aim to give a framework for future privacy-preserving cross-domain recommender system research. Their construction protects the rating values, but the recommender systems know the information that a user rated a specific item. Another issue with their design stems from the generation of the encryption key pair. Matrix factorization operations are through homomorphic operations, and the output is the encrypted latent vectors. Therefore the dot product that the users receive as the rating prediction is encrypted. In order to understand the generated rating, the users request the private key per recommendation. A user holding the secret key can collect information over the network to see the ratings generated for other users, which is still an issue in the semi-honest setting. Additionally, the authors do not consider the overhead of refreshing the keys in use or the other issues with sharing the private key with end-users.

In the implementation paper [25], they compare their method in terms of runtime and communication overhead with [24] and [20]. They test their design on several synthetic and real datasets. However, due to the incompleteness of the underlying cross-domain recommender model, they cannot provide any predictive performance metrics. In terms of communication and runtime performance, they are worse than [20] but better than [24].

Chen et al. [6] propose a differentially private rating sharing system between the two domains to achieve a cross-domain task. They propose publishing the source domain rating matrix using differential privacy on the rating values. They employ an auto-encoder to learn the user embeddings, from which they reconstruct the source domain rating matrix and measure the loss. For the cross-domain recommendation task, they use the user embeddings learned from the perturbed source domain to discover the user latent factors in the target domain. Their construction aims at preserving the source domain privacy from the target domain, but the recommender system knows the target domain ratings. Similarly, the source domain possesses the original rating matrix and performs perturbation according to the rating values. Overall, in their proposed system, users' rating privacy is not preserved against the recommender systems they belong to, as their privacy requirements differ from ours.

The design proposed by **Yu et al.** [37] is the latest work in privacy-preserving cross-domain recommenders currently, to the best of our knowledge. Their use case transfers information from the more general e-commerce domain to the recommendation of healthcare wearables such as smartwatches. Similar to [6], they assume the domains have access to submitted plaintext rating information. They propose sharing the user latent factors between the domains and keeping item latent factors hidden; therefore, the target domain cannot discover the submitted ratings by the source domain users. Differently than [6], this scheme of sharing the latent factors preserves the information that a given user rated an item. However, their privacy requirements are defined differently from our research problem, as the ratings are public against the recommender systems.

Designing a cross-domain recommender system with rating privacy remains an open research field. Table 3.2 summarizes the privacy-preserving cross-domain recommender system designs.

Table 3.2: Summary of Privacy-Preserving Cross-Domain Recommender Systems

Design	Rating Privacy	Interdomain Privacy	Overlap	Predictive Performance Loss	Runtime Cost
Ogunseyi et al.[26, 25]	Yes	Yes	Full	Not given	High
Chen et al. [6]	No	Yes	Full	Low	Low
Yu et al.[37]	No	No	Full	Low	Low

4 Privacy Preserving Matrix Factorization

Matrix factorization remains a ubiquitous technique in recommender systems, functioning as the cornerstone of single-domain and cross-domain recommendation schemes. As we discuss in Section 3.2, designing a matrix factorization scheme that is privacy-preserving is an active research topic. This chapter presents two approaches for designing a privacy-preserving matrix factorization scheme. We aim to provide a design with formal privacy guarantees that existing recommender systems can adopt without incurring major runtime and communication costs or significant predictive performance loss.

We consider a setting with users and the recommender system. In our setting, the recommender system refers to two non-colluding parties carrying out independent computations over the rating information they receive from the users. Multiple parties in a semi-honest setting increase the model's accuracy compared to single-party privacy-preserving approaches with noise addition. These two parties can be two computational servers hosted by two independent cloud service providers to prevent colluding in the real-world implementation. Spreading the computation over noise-added data to multiple servers allows us to reduce performance loss while maintaining privacy.

The privacy requirements for our system are as follows:

- The item associated with a submitted rating is private. However, the fact that a user rated a particular item leaks information on the user as shown in [35]. Therefore, the rated items should not be distinguishable by the recommender system.
- The values of ratings are private information. The recommender system components should not be able to differentiate between users based on the submitted rating.

We discuss the protocol design details in Section 4.1. Section 4.2 discusses the implementation details. We present results, complexity and performance analysis in Section 4.3.

Symbol	Definition
U	The user set
V	The item set

Symbol	Definition
u	User id
i	Item id
$R\left(R^1, R^2\right)$	Rating matrix (Received by Party 1 and Party 2)
$r_{u,i}$	Submitted rating of user u on item i
$\widehat{r}_{u,i}$	Predicted rating of user u on item i
$\bar{r}_u$	Average rating of user u
$\mathbf{U}\left(\mathbf{U}^1, \mathbf{U}^2\right)$	The user latent factor matrix (Computed by Party 1 and Party 2)
$\mathbf{V}\left(\mathbf{V}^1, \mathbf{V}^2\right)$	The item latent factor matrix (Computed by Party 1 and Party 2)
b_u^1, b_u^2	User biases calculated by Party 1 and Party 2
b_i^1, b_i^2	Item biases calculated by Party 1 and Party 2
μ^1, μ^2	Global mean calculated by Party 1 and Party 2
sk	Hash key held by users
$HMAC(sk, i)$	Keyed hash of item i

4.1. Design

There is a one-time initialization phase before the users can send their ratings to the recommender system. We employ a trusted third party to generate a secret key for the keyed hash function. This secret key is distributed to the patients who want to submit ratings for their physicians. The secret key allows the patients to generate the same hash for a given physician using a keyed hash function as $HMAC(sk, i)$. Instead of submitting ratings with item identifiers, the users submit ratings with the hash of the identifiers. The key-generating trusted third party and the users do not collude with the recommender system servers (Party 1 and Party 2). This construction allows us to address the first privacy concern, as the parties cannot link the generated hashes with the publicly available physician identifiers. However, the users can regenerate the hashes of the physician identifiers upon receiving the predicted ratings using the secret key they have, and they can identify the recommendation per physician.

We present two privacy-preserving matrix factorization schemes addressing the second privacy concern. In our schemes, the parties perform Singular Value Decomposition operations over the plaintext rating data to preserve runtime performance while gaining on the privacy aspect. For this purpose, we use differential privacy and multi-party computation. We have two main approaches to maintaining predictive performance: increasing the redundancy of the operations over noise masked data and splitting the noisy data into additive shares.

4.1.1. Increasing the Redundancy

The main idea behind this construction is to alleviate the negative effects of the differential privacy noise by introducing a multi-party setting and increasing the redundancy of the matrix factorization operations. The introduced parties are assumed to be honest but curious and do not collaborate. We select the number of non-colluding parties executing recommendations as two for simplicity in the experiments and the design. However, the number of parties can be greater to increase accuracy without loss of generality.

After deciding on the rating and the associated physician, the user samples two fresh noise instances from Laplacian distribution $Lap\left(\frac{\Delta f}{\epsilon}\right)$ where the sensitivity value Δf is the difference between the maximum and the minimum rating as:

$$\begin{aligned} noise_1 &\sim Lap\left(\frac{\Delta f}{\epsilon}\right), \\ noise_2 &\sim Lap\left(\frac{\Delta f}{\epsilon}\right). \end{aligned} \tag{4.1}$$

The user masks the rating as:

$$\begin{aligned} r^1_{u,i} &= r_{u,i} + noise_1\,, \\ r^2_{u,i} &= r_{u,i} + noise_2\,. \end{aligned} \tag{4.2}$$

The user also generates the hash of the physician id using the key and sends it to the two parties as:

$$\begin{aligned} Party1 &: (u, r^1_{u,i}, HMAC(sk, i))\,, \\ Party2 &: (u, r^2_{u,i}, HMAC(sk, i))\,. \end{aligned} \tag{4.3}$$

It is important to note that a fresh noise is created per party receiving the rating. Upon receiving the masked rating values, the parties perform SVD [21] algorithm over the values until the model converges.

Upon receiving a request the request for a recommendation from user u, the parties separately generate the recommendation for all $i \in V$ as:

$$\begin{aligned} \hat{r}^1_{ui} &= b^1_u + b^1_i + \mu^1 + (\mathbf{U}^1_u)^\mathsf{T} \cdot \mathbf{V}^1_i\,, \\ \hat{r}^2_{ui} &= b^2_u + b^2_i + \mu^2 + (\mathbf{U}^2_u)^\mathsf{T} \cdot \mathbf{V}^2_i\,. \end{aligned} \tag{4.4}$$

Recall from Chapter 2.1; we use the model outputs, user biases, item biases, global mean, and item and user latent factors of the SVD algorithm to generate the prediction.

The user receives the predicted ratings from the two parties and takes the element-wise **average** of the predicted values to get $\hat{r}_{ui}$ for all $i \in V$ as:

$$\hat{r}_{ui} = \frac{\left(\hat{r}^1_{ui} + \hat{r}^2_{ui}\right)}{2}\,. \tag{4.5}$$

To check which item the predicted rating belongs to, the user takes the hash of the items they are interested in and matches the received ratings.

Furthermore, we present another way of noise addition for a higher predictive performance where the two freshly created noise values are of opposite signs. One party gets the masked rating matrix using only positive noise, while the other gets the matrix masked with negative noise. This can be noted as:

$$\begin{aligned} noise_1 &\sim Lap\left(\frac{\Delta f}{\epsilon}\right), \\ noise_2 &\sim Lap\left(\frac{\Delta f}{\epsilon}\right), \end{aligned} \tag{4.6}$$

$$\begin{aligned} r^1_{u,i} &= r_{u,i} + |noise_1|\,, \\ r^2_{u,i} &= r_{u,i} - |noise_2|\,. \end{aligned} \tag{4.7}$$

Through experiments, we discuss the effect of using noise with opposite signs in the evaluation section.

4.1.2. Additive Separation

The main idea behind this construction is to achieve a lower sensitivity value to the differential privacy noise added to the ratings. This idea is achieved by dividing the rating by two and using the generated noise over the two additive shares so that their addition results in the initial rating. The sensitivity value Δf for the added noise is half the possible range of the data received by the parties. The additive nature of dot product operation allows for factorization over additive shares. As per the redundancy approach, we select the number of non-colluding parties executing recommendations as two for simplicity, but it can also be greater.

In this construction, the user divides their rating by two and generates one instance of noise using Laplacian distribution. Then, they add the generated noise to one of the shares, subtract the same noise value from the other share, and then send the shares to the parties as:

$$noise \sim Lap\left(\frac{\Delta f}{\epsilon}\right), \tag{4.8}$$

$$\begin{aligned} r^1_{u,i} &= \frac{r_{u,i}}{2} + noise, \\ r^2_{u,i} &= \frac{r_{u,i}}{2} - noise. \end{aligned} \tag{4.9}$$

Received vales are

$$\begin{aligned} &Party1 : (u, r^1_{u,i}, HMAC(sk, i)), \\ &Party2 : (u, r^2_{u,i}, HMAC(sk, i)). \end{aligned} \tag{4.10}$$

It is important to note that the noise is generated once unlike the previous approach and $r^1_{u,i} + r^2_{u,i} = r_{u,i}$. Upon receiving the masked rating values, the parties perform the Funk SVD algorithm over the values until the model converges. The existing recommender systems that perform the SVD algorithm require no modification with our scheme. The matrix factorization operations are identical to the non-privacy preserving implementations.

Upon receiving a request for a recommendation from user u, the parties separately generate the recommendation for all $i \in V$ with:

$$\begin{aligned} \hat{r}^1_{ui} &= b^1_u + b^1_i + \mu^1 + (\mathbf{U}^1_u)^\mathsf{T} \cdot \mathbf{V}^1_i, \\ \hat{r}^2_{ui} &= b^2_u + b^2_i + \mu^2 + (\mathbf{U}^2_u)^\mathsf{T} \cdot \mathbf{V}^2_i. \end{aligned} \tag{4.11}$$

Recall from Chapter 2.1; we use the model outputs, user biases, item biases, global mean, and item and user latent factors of the SVD algorithm to generate the prediction.

The user receives the predicted ratings from the two parties and takes the element-wise **summation** of the predicted values to get $\hat{r}_{ui}$ for all $i \in V$ as:

$$\hat{r}_{ui} = \left(\hat{r}^1_{ui} + \hat{r}^2_{ui}\right). \tag{4.12}$$

To check which item the predicted rating belongs to, the user takes the hash of the items they are interested in and matches the received ratings.

4.2. Implementation

In order to test the performance of the privacy-preserving matrix factorization compared to existing solutions, we provide a proof-of-concept implementation of our design. In addition to our designs with multiple parties performing operations, we implement an experiment with one factorization operation over noise added rating matrix as the naive way of applying differential privacy in a single-party setting. This additional experiment tests whether increasing operations' redundancy improves predictive performance.

We use an existing Funk SVD library implemented in Python for the experiments, which is an optimized version of the Funk SVD algorithm[1], implemented using Numba. The learning rate is 0.001, and the regularization parameter is 0.01. Through trial and error, we choose these parameters to allow the noise-added models to converge in a given iteration constraint. In addition, we use varying factors during the experiments for a more in-depth analysis of the effect of the number of factors on predictive performance.

As the dataset, we use MovieLens 100k, a dataset by the Grouplens Research Project at the University of Minnesota. It consists of 100,000 ratings from 943 users and 1682 items [2]. We use this dataset to compare other state-of-the-art privacy-preserving matrix factorization and non-privacy preserving solutions. As with previous works, we employ a train-test split of 80/20 and do not fix the split across experiments. The ratings are from 1 to 5, which sets the sensitivity of the Laplacian noise to be 4 in 4.1.1 and 2 in 4.1.2 as the halved ratings are from 0.5 to 2.5. We used different epsilon values (0.1, 0.3, 0.5, 0.7, 1, 1.5, 2, 3) during the experiments to better understand the effect of privacy on predictive performance.

In order to simulate the Increased Redundancy approach, we duplicate the training matrix and generate noise using the Laplacian distribution.

We mask the duplicated rating matrices additively using

$$\begin{aligned} [noise_1]_{ui} &\sim Lap\left(\frac{4}{\epsilon}\right), \\ [noise_2]_{ui} &\sim Lap\left(\frac{4}{\epsilon}\right), \end{aligned} \tag{4.13}$$

and finally as:

$$\begin{aligned} R^1 &= R + [noise_1], \\ R^2 &= R + [noise_2]. \end{aligned} \tag{4.14}$$

When the two models converge, we element-wise average the biases, the latent factors, and global means. Finally, we generate the predictions using the combined outputs over the test set.

Additive separation has a similar approach in implementation. For this, we also duplicate the training matrix and generate training data for the parties as:

$$[noise]_{ui} \sim Lap\left(\frac{2}{\epsilon}\right), \tag{4.15}$$

$$\begin{aligned} R^1 &= \frac{R}{2} + [noise], \\ R^2 &= \frac{R}{2} - [noise]. \end{aligned} \tag{4.16}$$

Although to simulate the addition operation on the user side, we sum the biases and the global means. Then, we concatenate the latent factors for corresponding items and users. Due to the dot product operation in the prediction stage, this concatenation operation has the effect

[1]https://github.com/gbolmier/funk-svd

[2]https://grouplens.org/datasets/movielens/100k/

of addition. Finally, similar to the previous experiments, we combine the outputs of the parties to generate the predictions faster for experiments.

We track metrics such as mean absolute error(MAE) and root mean square error(RMSE) across different epsilon values and dimensions of latent factors. We also note the time spent for the models to converge and for the prediction generation to better understand the usability of the privacy-preserving solutions.

We run the experiments on a PC with an Intel(R) Core i5-8250U Windows 10, at 1.60GHz CPU with 8GB RAM, and use Python 3.7 as the programming language. In both experiments, the models of Party 1 and Party 2 converge independently, simulating the non-collaborating parties of the design. Our proof of concept implementation and the adversary simulation are available on `https://github.com/aslikarahan/CDRS-with-privacy`.

4.3. Evaluation

4.3.1. Predictive Performance

In this section, we present the results of our experiments.

Our first point of comparison is in Figure 4.1, showing the mean absolute error values for four epsilon values and different schemes. The single-party construction refers to the naive way of adding differential privacy noise to the rating matrix and performing matrix factorization. The two-party construction is the approach from Section 4.1.1 where two parties perform matrix factorization of the two noisy rating matrices they received. ”Without noise” is the single party setting without privacy, constituting a baseline performance.

First, the relationship between the latent factor dimension and error is different between privacy-preserving schemes and the baseline without noise. We see a gradual increase in error until the dimension of 50, and then the model accuracy stabilizes. This phenomenon is because the rating matrix is sparse and the lower dimension for latent factors captures the variability better. The higher number of latent factors results in overfitting on *noisy data*, which explains the gradual increase in error in privacy-preserving schemes. The baseline also shows a minimal increase due to overfitting, but the increase is minor since overfitting happens on the actual data and not noisy data.

The second observation is a clear trade-off between privacy and accuracy. Smaller epsilon values provide higher levels of privacy, the added noise is greater, and the model performs worse. Smaller epsilon values are not feasible in this scheme, as an average error of 2 is unacceptable in a rating range of 5. However, the recommender systems can decide this trade-off concerning their requirements on accuracy and privacy.

The third observation is that increasing the redundancy of operations over differentially private rating matrices allows the recommender systems to reduce the effects of noise addition. The two-party setting outperforms the single-party setting over noise-added data in every epsilon value. Increasing the operations over noisy data does not result in noise accumulation but rather an increase in predictive performance with noise counterbalancing. We see the same trends with the RMSE metric in Figure 4.2.

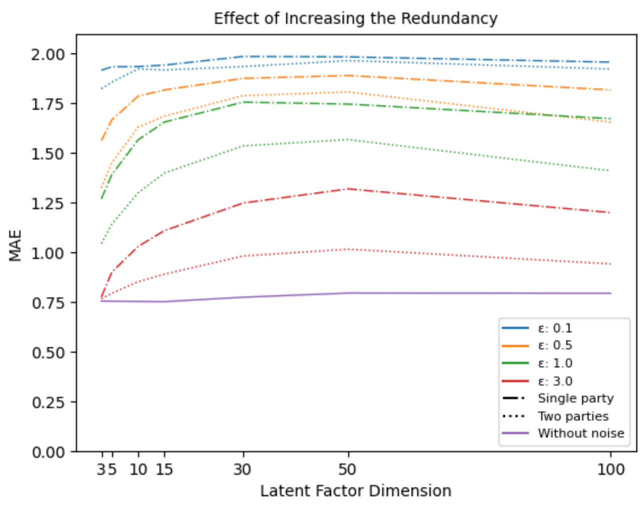


Figure 4.1: Mean Average Error with respect to latent factor dimension

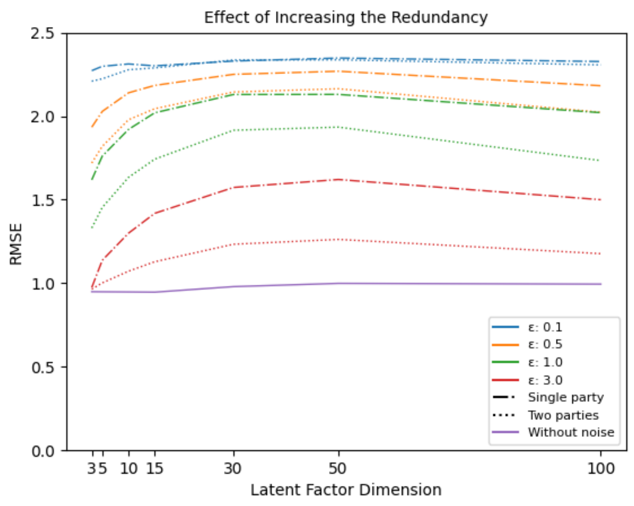


Figure 4.2: Root Mean Squared Error with respect to latent factor dimension

Furthermore, we propose a particular way of adding noise where one server gets the positive signed noise, and the other gets the negative signed noise, as given in 4.1.1. We see a better predictive performance if the added noise is of opposite signs for the two parties. The average of the rating predictions learned through opposite signed noisy ratings approximates the original prediction better. Furthermore, adding noise of the opposite signs outperforms the naive way of increasing the redundancy in both metrics, as seen in Figure 4.3 and 4.4, while ensuring the same formal privacy guarantee.

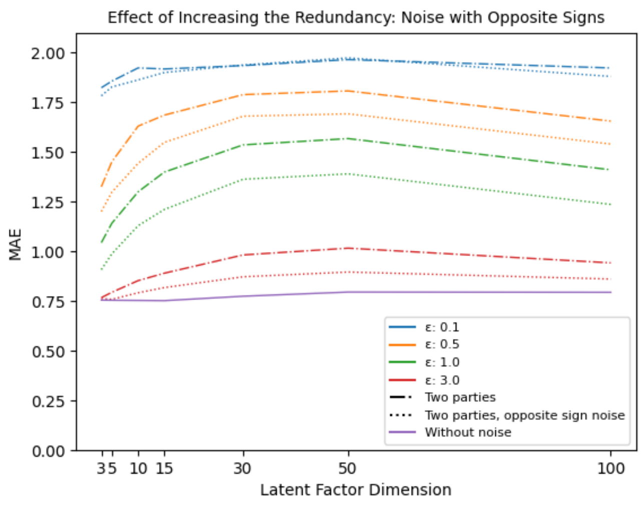


Figure 4.3: Mean Average Error with respect to latent factor dimension, noise with opposite sign

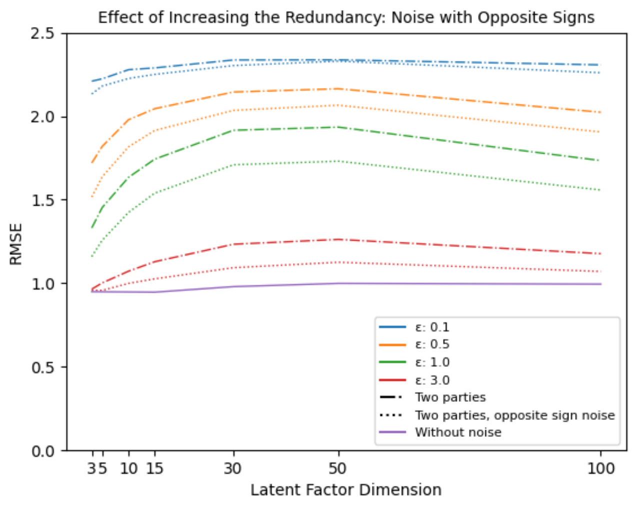


Figure 4.4: Root Mean Squared Error with respect to latent factor dimension, noise with opposite sign

Finally, we evaluate the effectiveness of our second approach of additive separation in Section 4.1.2. Recall that additive separation refers to splitting the rating matrix into two parts with differential privacy noise that adds to the original matrix. We see a drastic improvement over the single-party privacy-preserving setting of naive noise addition. This improvement is because the sensitivity value is half as the ratings are half, which allows us to achieve the same privacy guarantee with smaller noise. We see this significant performance improvement in both MAE and RMSE, in Figures 4.5 and 4.6.

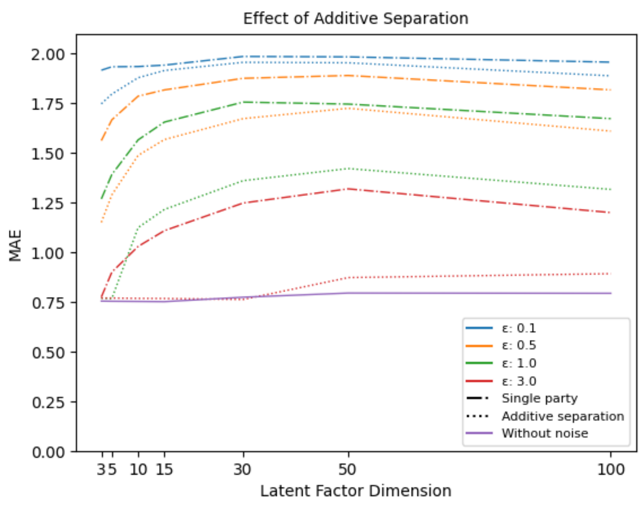


Figure 4.5: Mean Average Error with respect to latent factor dimension, additive separation

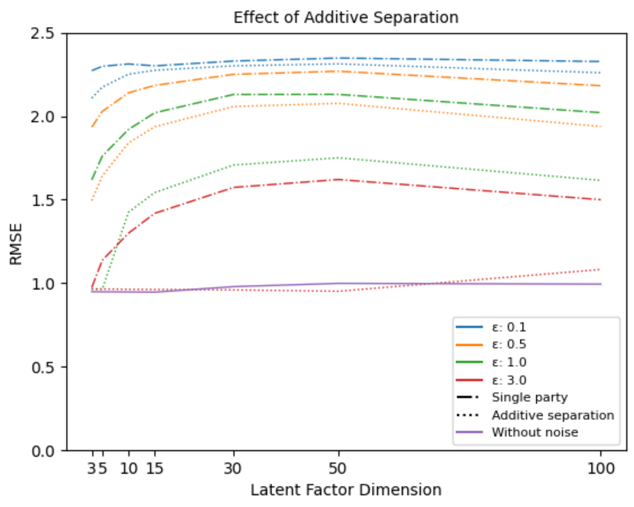


Figure 4.6: Root Mean Squared Error with respect to latent factor dimension, additive separation

Finally, we compare both approaches of privacy-preserving matrix factorization in Figure 4.7 and 4.8 with the baseline. Again, we see the privacy and accuracy trade-off, as the higher epsilon values result in lower errors. The additive separation approach performs best, resulting in the same predictive performance at $\epsilon = 1.5$. Overall, the additive separation design outperforms all other privacy-preserving factorization methods we propose in this work.

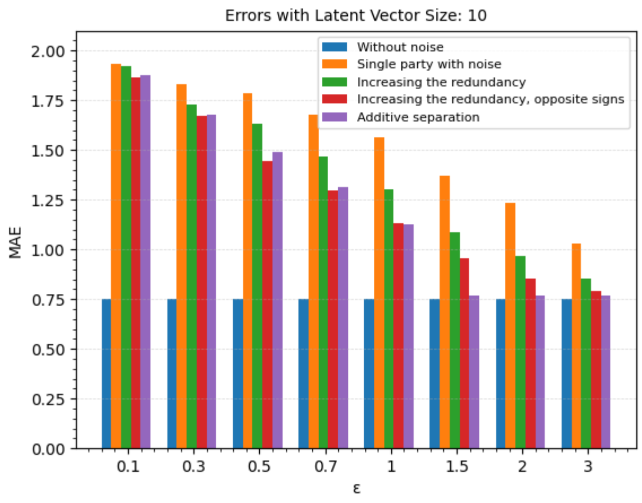


Figure 4.7: Mean Average Error Comparison of all PPMF designs

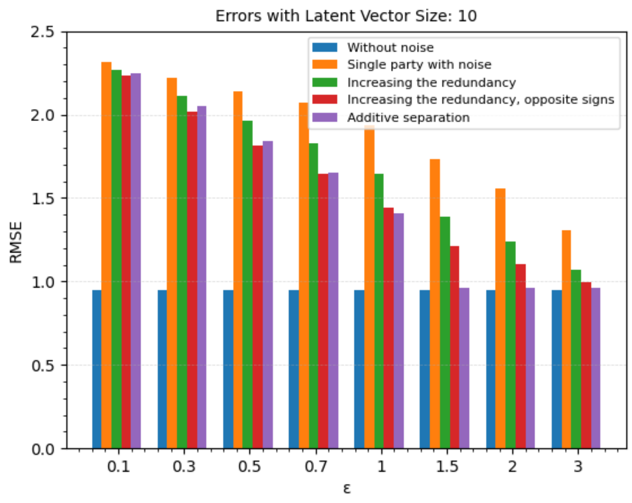
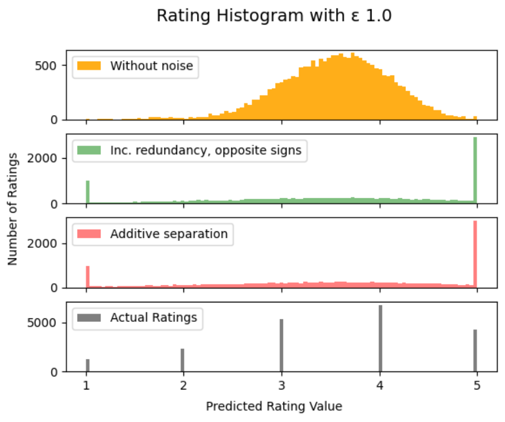
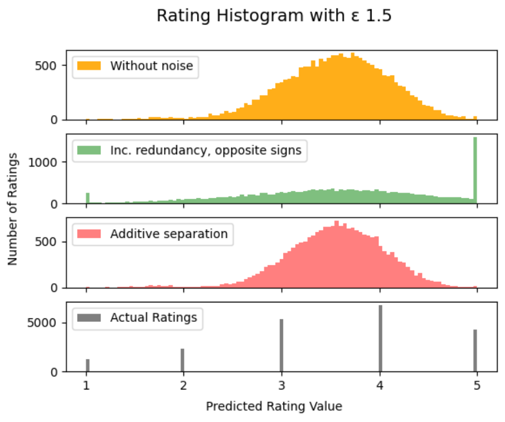


Figure 4.8: Root Mean Squared Error Comparison of all PPMF designs

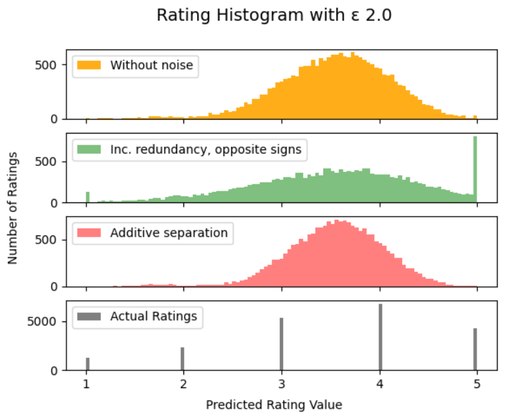
In order to give a better insight into the predictions generated by the privacy-preserving matrix factorization methods, we present histograms of the ratings in Figure 4.9. In the histograms, we show the predictions of the best-performing privacy-preserving methods. We see how the added noise affects the distribution of the predictions; the SVD algorithm generates almost uniform predictions in the given rating space when the added noise is too high. When the added noise is bigger, the predictions are spread out. The algorithm clips the predicted ratings to be between 1 and 5; therefore, we see an uneven number of predictions of 1 and 5. The privacy-preserving methods output a similar rating distribution as the noise gets smaller.



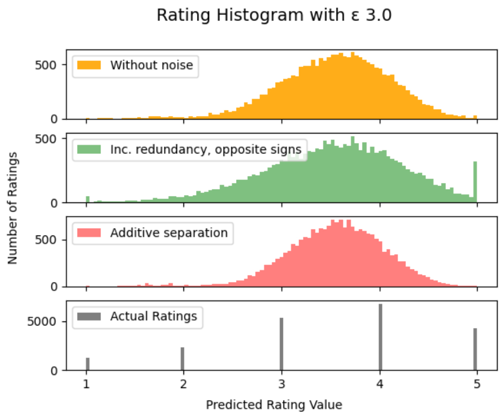
(a) 1a



(b) 1b



(c) 1b



(d) 1b

Figure 4.9: Rating Prediction Histograms

4.3.2. Runtime

In this section, we discuss the runtime implications of our design. We measure the total elapsed time running an experiment. More specifically, we measure the time it takes to generate predictions for all latent factor dimensions (3, 5, 10, 15, 30, 50, 100) per epsilon value. The Funk SVD implementation is an optimized library that generates the recommendations in seconds (42 seconds for 20 million ratings). Since we use the 100k dataset, the differences in runtime get smaller. Therefore, we measure the time to run experiments over all the latent factor dimensions to observe differences better.

Table 4.2: Runtime in seconds per Epsilon

	0.1	**0.3**	**0.5**	**0.7**	**1**	**1.5**	**2**	**3**
Without noise	30.5							
Single party with noise	19.4	32.2	36.1	31.2	30.5	30.8	40.2	34.6
Increasing the redundancy	29.9	62.1	60.9	60.3	61.4	58.2	59.1	59.2
Increasing the redundancy, opposite signs	50.4	62.2	58.7	57.8	57.3	57.9	58.8	57.4
Additive Separation	60.8	59.7	58.7	58.4	59.1	54.3	51.0	36.3

In Table 4.2, we see that the added noise does not hurt the runtime performance significantly compared to the non-privacy preserving version. We also see the total time doubles in the multi-party setting, but this does not reflect the real-world performance. As the recommender system parties are independent, they can run the matrix factorization simultaneously. The outliers in the runtime are due to the early stopping feature of the implementation. When the model is not improving significantly, the gradient descent process concludes before reaching the maximum iteration. Among the privacy-preserving schemes, we see that the best performers, oppositely signed noise and additive separation, converge slightly faster than the single party with noise and the naive way of increasing redundancy. This is likely because those schemes preserve the rating structure more than the others, allowing the models to converge faster.

Overall, the noise addition does not hurt the runtime significantly from the recommender system point of view. The hashes they receive as item identifiers do not affect the matrix factorization operation. Thus, our scheme does not incur an additional runtime cost compared to non-privacy preserving single-domain recommender systems. From the user's point of view, the physician identifier hash generation is an offline phase, not affecting the runtime. Additionally, combining the prediction on the user side is $\mathcal{O}(n)$, linear with the number of items.

4.3.3. Communication Overhead

Compared to the non-privacy preserving single-domain recommender systems, our design has a higher communication overhead for both approaches. The first communication overhead is the hash key distribution and user registration, as we have a communication step between the user and the trusted key-generating third party for every user joining the system. The communication is doubled in the rating submission stage, as the user sends their rating to two entities instead of one. The most significant difference in communication overhead is the ability of the recommender system to provide a single item prediction. In non-privacy preserving versions, the user can ask for a prediction on a specific item, as the items associated with the ratings are known. In our design, the user receives predictions of all items at once. However, the user can also request all predictions in non-privacy settings, making this difference in communication overhead unrelated to the privacy addition.

4.3.4. Privacy Analysis

For the first privacy requirement of our system, we aim to hide the associated item information of a rating submission. As we stated at the beginning of this chapter, the information that a person rated an item can identify the person or at least learn information about them. In the healthcare setting, this risk is more significant as physicians have defined specialties that are public information. Therefore, a rating associated with a specific physician narrows down a patient's possible ailments, hurting their privacy.

We employ a keyed hash function to hide item information in our design. Cryptographic hash functions are one-way functions; as such, the parties receiving the hash of the item identifier cannot revert the function to obtain the item identifier. Additionally, cryptographic hash functions are preimage resistant. Therefore, finding a message that results in the same hash is infeasible in a given time and computational power. In our use case, the message space is restricted to physician identifiers. This means low entropy; it allows the parties to brute force in a restricted message input space. However, our keyed hash function's key component adds randomness to the input space, rendering it infeasible for anyone without the correct key to brute force identifiers. As long as the key holders and the key generating external party do not share their hash key with the recommender system parties, our design achieves item privacy.

For the second requirement, colloquially ”rating privacy”, we present how different epsilons provide different privacy levels. In order to assess the effect of the noise magnitude on the rating privacy, we implement a rudimentary adversary that attempts to guess the ratings of individuals based on the noisy rating it receives.

It is public information that the actual rating values are between 1 and 5. However, with the added Laplace noise, the noisy ratings fall outside the 1-5 range. The adversarial guessing is as follows: the recommender system clips the noisy ratings into the 1-5 range and assumes the ratings are what it receives. The parties receive half of the rating plus or minus Laplacian noise for the additive separation approach. Therefore, to guess the actual ratings, the adversary multiplies the noisy rating they receive and clips the result into the 1-5 range. Finally, we average the reconstruction error to give a complete overview of the system for the additive separation and increased redundancy approaches in two-party settings.

With this adversary implementation, we provide insight into how the epsilon value correlates to the privacy of the ratings against the recommendation server. The error in guessing the rating corresponds to the privacy level achieved by design. For this assessment, we use MAE as before as it provides interpretable numeric values in the rating range. Higher guessing error for the parties suggests higher user privacy, as the parties cannot recover the ratings successfully from the noisy inputs. The errors are in Figure 4.10. We observe that the parties can guess the user ratings better as the ϵ value gets bigger and the added noise gets smaller.

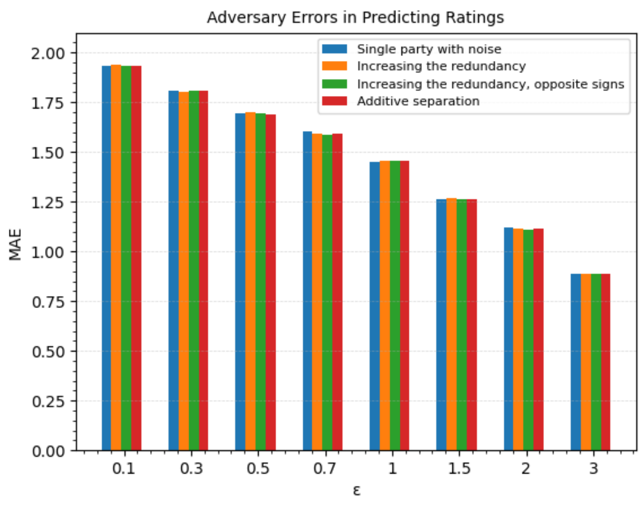


Figure 4.10: Adversary Errors in Predicting Actual Ratings

It is important to note that in the additive separation scheme we present, parties must be non-collaborating. When parties collaborate, they can reveal the exact rating by adding the shares they hold. In the increased redundancy approach, however, the case is different. The parties collaborating will not result in the exact rating. To uncover the private ratings, they can average their noisy ratings to approximate the actual rating, but it will not be exact. In the increased redundancy approach with the oppositely signed noise, the ratings' average better approximates the rating.

To empirically confirm this intuition, we implement a proof of concept and plot the errors in guessing where the parties collaborate. The results are in Figure 4.11. As expected, in the additive separation approach, user privacy is not preserved when the parties collaborate. The opposite sign approach is more vulnerable to privacy loss when parties collaborate than the regular increased redundancy approach and the case with a single party.

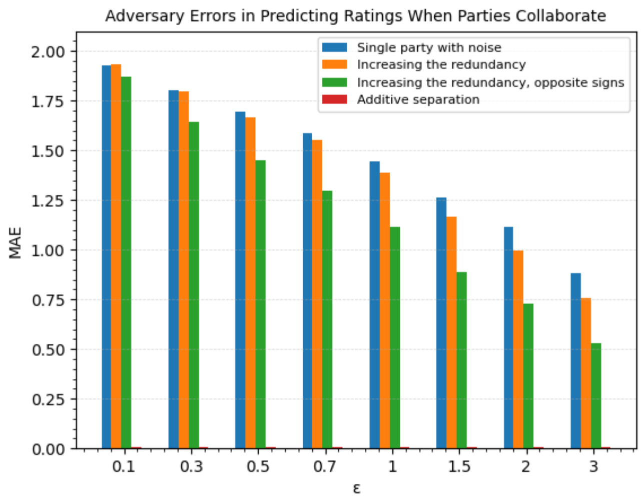


Figure 4.11: Adversary Errors in Predicting Actual Ratings When Parties Collaborate

Overall, we see that privacy and accuracy constitute a trade-off in our Privacy-Preserving Matrix Factorization design with ϵ values that we test on providing a confusion between (1.9, 0.9) in a range of 5. Finally, the best predictive performers, increased redundancy with opposite signs and additive separation, are more vulnerable to privacy loss in cases of collusion, presenting another aspect of accuracy and privacy trade-off.

5 Privacy Preserving Cross Domain Recommendation System

Building on our privacy-preserving matrix factorization scheme in the previous chapter, we present our privacy-preserving cross-domain recommender system. As we discuss in Section 3.4, a research gap exists in designing a cross-domain recommender with rating privacy. In this chapter, we first present our recommender system design without privacy. Then we discuss how to introduce privacy to our design.

The cross-domain recommendation problem we focus on is to generate rating predictions for a new user using the ratings overlapping users between the domains. As per the recommender system terminology, we denote the domain where auxiliary information is transferred from as *source domain*. The *target domain* is where the recommendation task for the new user is carried out. Therefore we consider a setting with two domains, a varying number of overlapping users with interactions in both domains, and a cold start user with no interactions in the target domain. In the privacy-preserving version of our scheme, we also use a trusted third party to generate hash keys and a trusted third party to aid in secure dot product operations. Additionally, we use the matrix factorization method of additive separation discussed in the previous chapter. Therefore, each domain consists of two non-colluding parties: $Source_1$, $Source_2$, $Target_1$ and $Target_2$.The privacy requirements for our system are as follows:

- The item associated with a submitted rating is private. Therefore, the rated items should not be distinguishable by the recommender system.
- The values of ratings are private information. The recommender system components should not be able to differentiate between users based on the submitted rating.
- Identifiers of non-overlapping users between the domains should be kept private.
- The domain-specific ratings, as well as latent factor vectors of users, items, and other model outputs, are private, as domains can use them to recover the ratings

We present the design of the protocol without privacy in Section 5.1. Section 5.2 presents the privacy preserving version of our system. We discuss the implementation details in Section 5.3. We present results, complexity and performance analysis in Section 5.4.

5.1. Design without privacy

Symbol	Definition
U_s, U_t	The user set of the source or of the target domain
$U_{s,o}, U_{t,o}$	The overlapping user set at the source or target domain
$\mathbf{U}\left(\mathbf{U}^s, \mathbf{U}^t\right)$	The user latent factor matrix of the source or target domain
$\mathbf{U}_o\left({\mathbf{U}_o}^s, {\mathbf{U}_o}^t\right)$	The overlapping user latent factors of the source or target domain
$\mathbf{V}\left(\mathbf{V}^s, \mathbf{V}^t\right)$	The item latent factor matrix of the source or target domain
b_u^s, b_u^t	User biases of the source or the target domain
b_i^s, b_i^t	Item biases of the source or the target domain
μ^s, μ^t	Global mean of the source or the target domain
$r_{u,i}$	Submitted rating of user u on item i
$\widehat{r}_{u,i}$	Predicted rating of user u on item i
$\bar{r}_u$	Average rating of user u

Upon a recommendation request of a cold start user, the target domain initiates contact with the source domain to start the cross-domain recommendation procedure.

In this communication, the target domain shares the identifier of the new user U_{new} and the existing users U_t with the source domain. After receiving the identifiers, the source domain identifies the user overlap between the two domains. Then, the source domain performs the Funk SVD algorithm over the source rating matrix. Afterward, the source domain sends the user latent factors of the overlapping users $\mathbf{U}_o^s$ and the new user $\mathbf{U}_{new}^s$to the target domain, along with the mean rating of the new user $\bar{r}_{new}$.

The target domain performs the Funk SVD algorithm independently of the source domain over the target rating matrix. Upon receiving the user latent factors from the source domain, the target domain calculates two different kinds of similarity values over latent factors: inter-domain and intra-domain similarities. The design choice to calculate user similarity over latent factors instead of rating values is in the framework [14] developed by He et al., demonstrating that using latent factors results in better accuracy due to the sparsity reduction. Intra-domain similarity refers to the Cosine similarity of the new user to the overlapping users in the source domain, as:

$$sim\left(U_{new}, U_{s,o}\right) = \cos\left(\mathbf{U}_{new}^s, \mathbf{U}_o^s\right) . \tag{5.1}$$

This is where the previous source domain experience of the new user is considered. Inter-domain similarity is calculated over the overlapping users' latent factors from the source domain and the target domain using Cosine similarity as:

$$sim\left(U_{s,o}, U_{t,o}\right) = \cos\left(\mathbf{U}_o^s, \mathbf{U}_o^t\right) . \tag{5.2}$$

The new user has no interactions in the target domain yet, so this calculation does not include the new user. Calculating the inter-domain similarity is a domain adaptation strategy that aims to adjust for the overlapping users whose behavior may vary across different domains.

The similarities of the new users with the overlapping users in the target domain are calculated by multiplying the corresponding inter-domain and intra-domain similarities as:

$$sim\left(U_{new}, U_{t,o}\right) = sim\left(U_{new}, U_{s,o}\right) \cdot sim\left(U_{s,o}, U_{t,o}\right) . \tag{5.3}$$

The target domain uses the following equation to generate the predictions for the new user for a given item j:

$$\widehat{r}_{new,j} = \bar{r}_{new} + \frac{\sum_{u_{t,o} \in U_{t,o}} \left(sim\left(U_{new}, u_{t,o}\right) \cdot \left(r_{u_{t,o},j} - \bar{r}_{u_{t,o}} \right) \right)}{\sum_{u_{t,o} \in U_{t,o}} |sim(U_{new}, u_{t,o})|}. \tag{5.4}$$

It is important to note that in the cases where $r_{u_{t,o},j}$ does not exist, the target domain predicts the rating and uses the predicted $\widehat{r}_{u_{t,o},j}$ instead. The target domain uses the output of the Funk SVD algorithm to generate predictions.

Since this design has no privacy considerations, we can not meet any of the privacy requirements. Instead, the domains collaborate by sending information openly, giving each other full access to ratings, user, and item information.

5.2. Design with privacy

Following the cross-domain recommendation design for the new users in the previous section, we show how to make it privacy-preserving.

We employ the privacy preserving matrix factorization with the additive separation approach as detailed in 4.1.2. As there are two parties per domain performing the matrix factorization, the notation will be as follows:

Symbol	Definition
$\mathbf{U}\left(\mathbf{U}^{s1}, \mathbf{U}^{s2}\right)$	The user latent factor matrix of the source domain held by Party 1 and Party 2
$\mathbf{U}\left(\mathbf{U}^{t1}, \mathbf{U}^{t2}\right)$	The user latent factor matrix of the target domain held by Party 1 and Party 2
b_u^{s1}, b_u^{s2}	User biases of the source domain held by Party 1 and Party 2
b_u^{t1}, b_u^{t2}	User biases of the target domain held by Party 1 and Party 2
b_i^{s1}, b_i^{s2}	Item biases of the source domain held by Party 1 and Party 2
b_i^{t1}, b_i^{t2}	Item biases of the target domain held by Party 1 and Party 2
$\mu^{s1}, \mu^{s2}, \mu^{t1}, \mu^{t2}$	Global mean of the source or the target domain

We meet the first two privacy requirements using the previously detailed privacy-preserving matrix factorization method.

5.2.1. User overlap

In the non-privacy preserving version of the construction, the user overlap determination leaks the entire user identifier of the target domain to the source domain. In a privacy-preserving setting, source and target domains conduct a two-party private set intersection protocol to determine the overlapping and new users. In this protocol, the parties each have a set of identifiers, some overlapping, that they want to compute the intersection. The privacy aspect is that non-intersecting items of both parties are not revealed. They achieve this property by using the multiplication of identifier hashes together with encryption, and secure multiplication operations, as demonstrated by [7].

5.2.2. Similarity calculation

Contrary to the regular scheme, the domains do not hold entire latent vectors. Party 1 holds the first half of a given user latent factor vector without loss of generality, and Party 2 holds the second half. Similarities are demonstrated in Figure 5.1.

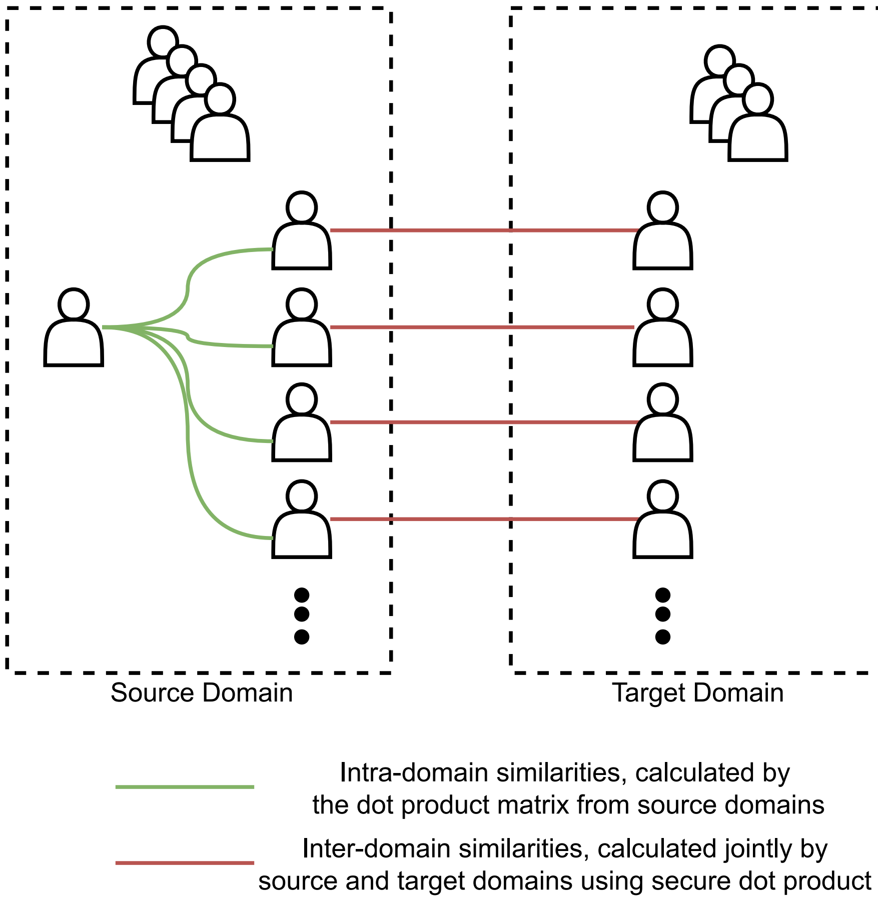

Source Domain
Target Domain
Intra-domain similarities, calculated by the dot product matrix from source domains
Inter-domain similarities, calculated jointly by source and target domains using secure dot product

Figure 5.1: Similarities in privacy preserving setting

Therefore, during the similarity calculation, $Target_1$ communicates with $Source_1$ to calculate inter-domain similarities as $Target_2$ communicates with $Source_2$. In order to calculate the cosine similarity between the overlapping users of the source and the target domains, we carry out the secure dot product protocol explained in Chapter 2.4. In the end, the parties do not end with shares of the secure dot product result. Instead, the target ends up with the dot products between overlapping users. Secure dot product protocol prevents latent factors from being leaked between the domains, addressing the privacy requirements.

For intra-domain similarity, source domains calculate the dot product between the new user and the overlapping users using the half of the latent vector they hold and send it over to the respective target domain parties. For the cosine calculation, the source domains also send the overlapping user vectors' dot products with themselves as the square of the magnitude of the latent factors. Below is the summary of the information held by the target domains by the end of this exchange:

$$\begin{aligned} Target_1 &: \left(\mathbf{U}_o^{s1} \cdot \mathbf{U}_o^{t1}\right), \left(\mathbf{U}_o^{s1} \cdot \mathbf{U}_o^{s1}\right), \left(\mathbf{U}_{new}^{s1} \cdot \mathbf{U}_{new}^{s1}\right), \\ Target_2 &: \left(\mathbf{U}_o^{s2} \cdot \mathbf{U}_o^{t2}\right), \left(\mathbf{U}_o^{s2} \cdot \mathbf{U}_o^{s2}\right), \left(\mathbf{U}_{new}^{s2} \cdot \mathbf{U}_{new}^{s2}\right). \end{aligned} \tag{5.5}$$

It is important to note that parties cannot deduce the latent factors from the above dot products as long as the dimension of the latent factors is greater than 2.

$Target_1$ and $Target_2$ exchange the dot products they have in addition to the $\left(\mathbf{U}_o^{t1} \cdot \mathbf{U}_o^{t1}\right)$ and $\left(\mathbf{U}_o^{t2} \cdot \mathbf{U}_o^{t2}\right)$. $Target_1$ and $Target_2$ perform summation over the dot products they receive from each other and use the dot products to calculate inter-domain and intra-domain similarities.

5.2.3. Prediction generation

After calculating the inter-domain and intra-domain similarities, both target domains calculate the overall similarity between the new user and the overlapping users in the target with multiplication as:

$$sim\left(U_{new}, U_{t,o}\right) = sim\left(U_{new}, U_{s,o}\right) \cdot sim\left(U_{s,o}, U_{t,o}\right). \tag{5.6}$$

We recall the rating prediction formula as:

$$\widehat{r}_{new,j} = \bar{r}_{new} + \frac{\sum_{u_{t,o} \in U_{t,o}} \left(sim\left(U_{new}, u_{t,o}\right) \cdot \left(r_{u_{t,o},j} - \bar{r}_{u_{t,o}}\right)\right)}{\sum_{u_{t,o} \in U_{t,o}} |sim(U_{new}, u_{t,o})|}. \tag{5.7}$$

The rating average of the new user $\bar{r}_{new}$ is present on the new user side as users keeping track of their average rating is a small overhead. Similarities are know by $Target_1$ and $Target_2$. The only missing values to generate the recommendations are the individual ratings (given or predicted) of the overlapping users $r_{u_{t,o},j}$ or $\widehat{r}_{u_{t,o},j}$ and the average ratings of the overlapping users $\bar{r}_{u_{t,o}}$. As we divide the ratings between $Target_1$ and $Target_2$, each has access to an additive share of these missing components. The denominator of the prediction formula is the same for both parties. The numerator has a summation operation that allows both parties to calculate it separately with the additive shares they possess and send the result to the requesting user to sum.

What is sent by the parties is as follows:

$$\begin{aligned} Target_1 &\longrightarrow \frac{\sum_{u_{t,o} \in U_{t,o}} \left(sim\left(U_{new}, u_{t,o}\right) \cdot \left(r^1_{u_{t,o},j} - \bar{r}^1_{u_{t,o}}\right)\right)}{\sum_{u_{t,o} \in U_{t,o}} |sim(U_{new}, u_{t,o})|}, \\ Target_2 &\longrightarrow \frac{\sum_{u_{t,o} \in U_{t,o}} \left(sim\left(U_{new}, u_{t,o}\right) \cdot \left(r^2_{u_{t,o},j} - \bar{r}^2_{u_{t,o}}\right)\right)}{\sum_{u_{t,o} \in U_{t,o}} |sim(U_{new}, u_{t,o})|}. \end{aligned} \tag{5.8}$$

Combination on the user side is

$$\begin{aligned} \widehat{r}_{new,j} = \bar{r}_{new} + &\frac{\sum_{u_{t,o} \in U_{t,o}} \left(sim\left(U_{new}, u_{t,o}\right) \cdot \left(r^1_{u_{t,o},j} - \bar{r}^1_{u_{t,o}}\right)\right)}{\sum_{u_{t,o} \in U_{t,o}} |sim(U_{new}, u_{t,o})|} + \\ &\frac{\sum_{u_{t,o} \in U_{t,o}} \left(sim\left(U_{new}, u_{t,o}\right) \cdot \left(r^2_{u_{t,o},j} - \bar{r}^2_{u_{t,o}}\right)\right)}{\sum_{u_{t,o} \in U_{t,o}} |sim(U_{new}, u_{t,o})|} \end{aligned} \tag{5.9}$$

5.3. Implementation

In order to test the baseline performance of our design and the cost of adding privacy, we implemented a proof-of-concept experiment setup. Unlike our previous matrix factorization implementation, we include a preprocessing step to simulate a source and target domain split. In the preprocessing step, we remove all items that received less than ten ratings and all users that contributed less than 20. This step makes our results comparable to other works with the exact preprocessing step.

To simulate the source and domain split, we take the steps given in [39] except for adjusting the sparsity between the domains. In their problem definition, source and target domains have low and high sparsity, respectively. However, our setting does not dictate such a difference in sparsity between the domains. Moreover, arguing for a difference between the domains for the hospital rating systems would have been incorrect. Therefore, we split the dataset into two domains with similar sparsity values in our implementation.

We first sample a number of users to be in the source domain. We select this number as 450, around half the total number of users in the MovieLens dataset (943 users). Next, we sample a number of users from the source domain to be the overlapping users between the domains. We denote the user overlap as percentages and explore 5%, 10%, 20%, and 30%. Although the authors explore higher percentages of overlap in the previous works, such as 50% and 70%, we assume the overlap percentage is lower in a hospital-patient setting. After denoting a set of users as overlapping users, we sample an additional number of users for the target domain. In the end, the target domain also has 450 users, including the overlap.

After splitting the source and target domain users, we split the items into two sets with no overlap. This situation aligns with our use case of physician ratings and hospitals being the domains, assuming that one physician can not work at multiple hospitals simultaneously. Our test set is the overlapping users' ratings in the target domain for the privacy-preserving and the regular scheme.

After splitting the dataset into source and target domains with partial user overlap and no item overlap, we first run the non-privacy version of our cross-domain construction. We run the FunkSVD algorithm over the source and target ratings with model parameters such as the learning rate of 0.001 and the regularization parameter of 0.01. we select each user in the set of overlapping users as the cold start user one by one, with the remaining overlapping users used for the prediction generation. We compare the predicted ratings of the overlapping users in the target domain and the actual ratings using mean absolute error(MAE) and root mean square error(RMSE) metrics. We find the number of factors that optimize the non-privacy preserving cross-domain predictions and use that dimension value in the privacy-preserving version.

In our proof of concept implementation, we selectively implement the sections of our design that affect the predictive performance of the system. As such, we exclude secure dot product operation and the item hashing process since they do not affect the prediction outcome of the design.

For the privacy-preserving version, we implement the two parties in source and target domains as $Source_1$, $Source_2$, $Target_1$ and $Target_2$. We carry out the privacy-preserving matrix factorization of Section 4.1.2 by dividing the ratings by two and sending the halved ratings masked with Laplacian noise to the respective parties of both domains. We run the FunkSVD algorithm with the parameters of the non-privacy preserving version and the optimized latent factor dimension. We then calculate the user similarities over the resulting latent factors and use the similarities to generate predictions as described in the previous section. Like the regular version, we compare the predicted ratings of the overlapping users in the target domain and the actual ratings using mean absolute error(MAE) and root mean square error(RMSE) metrics.

In our experiments, $Source_1$, $Source_2$, $Target_1$ and $Target_2$ converge independently, simulating the non-collaborating parties of the design. We run the experiments on a PC with an Intel(R) Core i5-8250U Windows 10, at 1.60GHz CPU with 8GB RAM, and use Python 3.7 as the programming language. Our cross-domain recommender implementation is available on `https://github.com/aslikarahan/CDRS-with-privacy`.

5.4. Evaluation

5.4.1. Predictive Performance

In this section, we present the results of our experiments.

Figures 5.2 and 5.3 show how different dimensions of latent factors affect the predictive performance in the cross-domain new user setting. We plot the MAE and RMSE of different latent factor dimensions. We see that the dimension of the latent factor does not significantly affect the recommendation performance, in line with the findings in [14]. Confirming the author's findings in the latent factor-based similarity model paper, we conclude that increasing the latent factor dimension after a certain threshold does not increase the predictive performance. Unlike our results in the matrix factorization section, we do not observe overfitting in this scheme.

Additionally, we see that 5% user overlap is not consistent in the predictive accuracy as the other overlap percentages. This inconsistency is most probably due to the size of our dataset, as the 5% user overlap corresponds to a test set of size 1393 in our experimental setup, and hence it is not generalizable. Aside from the 5% overlap case, increasing the user overlap results in a slightly lower error value.

We set the dimension as six during our comparison with the privacy-preserving version of the protocol. Following the visual inspection of the plots, we conclude that dimension 6 is where the predictive performance stops improving and is also the latent factor dimension corresponding to the elbow point in [14].

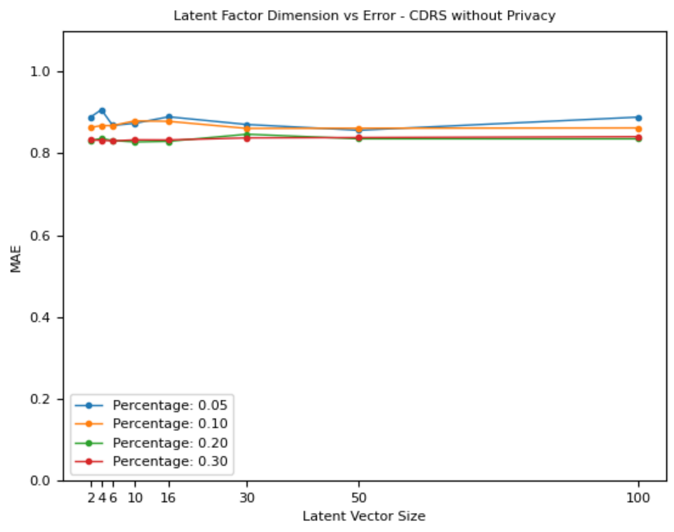


Figure 5.2: Mean Average Error with respect to latent factor dimension for different overlap values

Similar to our results in Section 4.3, we observe a trade-off between accuracy and privacy. However, the magnitude of the added noise has a lesser effect on the accuracy loss. This difference is because the construction relies on existing user ratings and similarity values. We can

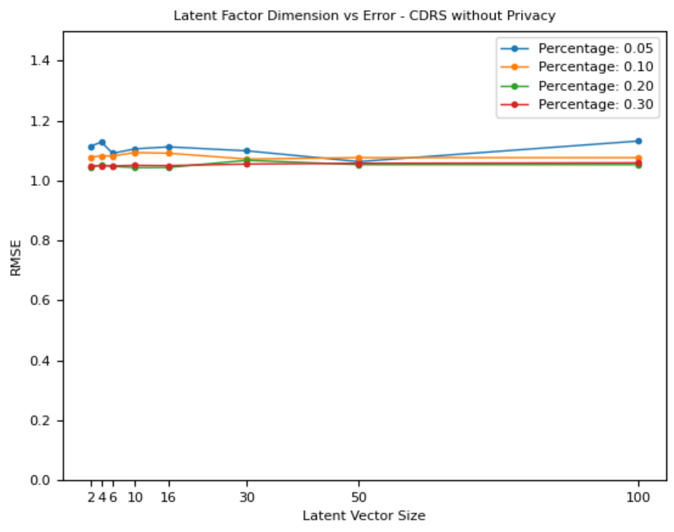
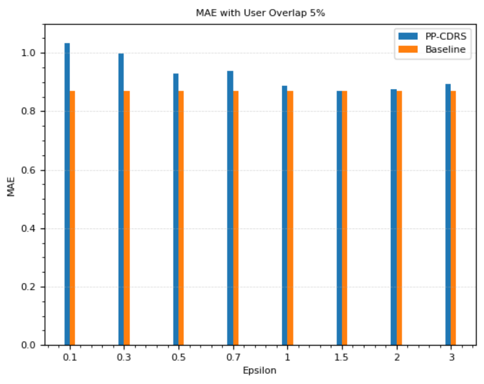


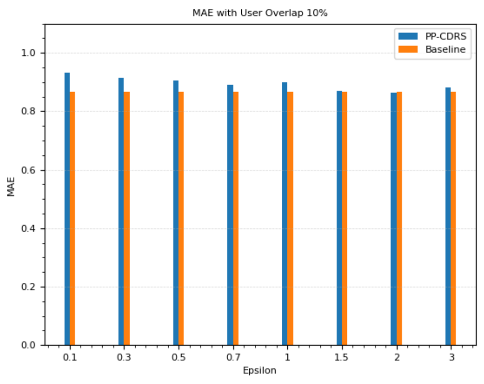
Figure 5.3: Root Mean Squared Error with respect to latent factor dimension for different overlap values

infer that the calculated similarities used in the prediction stage are small in magnitude, thus similar to a basic recommender system of user average. However, as we see the recommendation performance increasing with the increased user overlap percentage, it is not feasible to discard the similarity measure entirely.

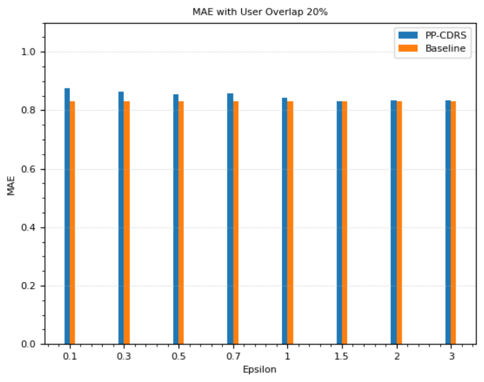
We see that with the increasing user overlap, the predictive performance is increasing. Overall, we conclude that learning user similarities over a differentially private rating matrix with a two-party setting and the additive separation approach detailed in Section 4.1.2 is a viable privacy-preserving cross-domain recommender design.



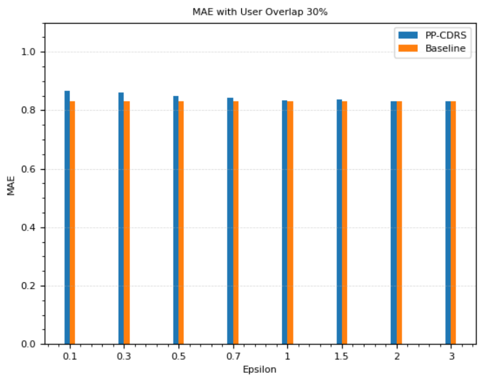
(a) MAE with User Overlap 5%



(b) MAE with User Overlap 10%

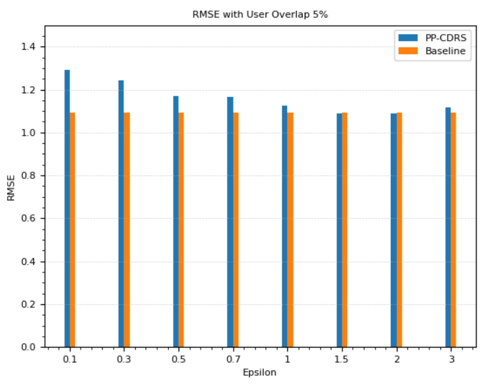


(c) MAE with User Overlap 20%

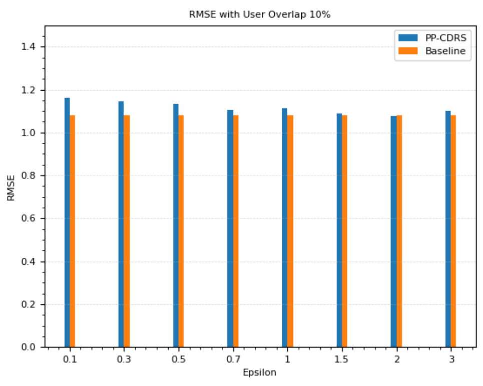


(d) MAE with User Overlap 30%

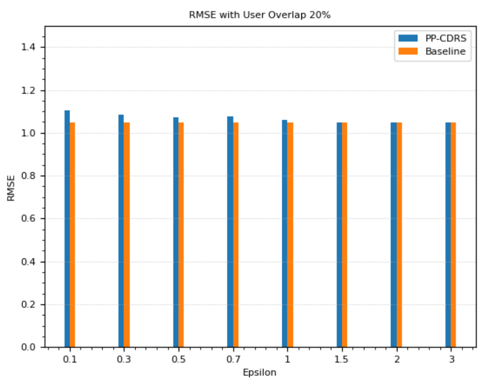
Figure 5.4: Mean Average Error comparison between non-privacy preserving and privacy preserving CDRS



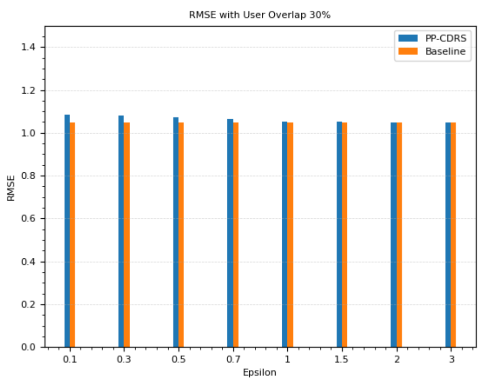
(a) RMSE with User Overlap 5%



(b) RMSE with User Overlap 10%



(c) RMSE with User Overlap 20%



(d) RMSE with User Overlap 30%

Figure 5.5: Root Mean Squared Error comparison between non-privacy preserving and privacy preserving CDRS

5.4.2. Runtime

As our proof of concept focused mainly on the effect of privacy on predictive performance, our experiments can not reflect the actual runtime performance. Another point is that we calculate the similarity between the new user and the remaining overlapping users in a new user setting. This calculation results in the runtime complexity of $\mathcal{O}(n)$ linear with the number of overlapping users. However, as a predictive performance-focused implementation, we rotate every user in the overlapping users set as the new user while generating test set predictions. Hence, the runtime complexity of our experimental setup is $\mathcal{O}(n^2)$ with respect to the number of overlapping users.

We can not provide elapsed time measurements as the size of the user overlap affects the test set size. However, in the real-life setting, the test set consists of all of the items in the target domain. Thus, the time complexity of a prediction generation is $\mathcal{O}(n * m)$ where n is the number of overlapping users, and m is the number of target items.

As demonstrated in the previous chapter, there is no significant difference between the runtime performance of non-privacy preserving and privacy-preserving matrix factorization schemes. Furthermore, as the two-party setting in source and target domains can be parallel, the matrix factorization phase does not incur additional runtime costs with the added privacy.

The effect of other aspects of our design, such as secure dot product and private set intersection on the runtime, can not be compared through our experimental setting.

5.4.3. Communication Overhead

In our design, the secure dot product and the similarity calculation generate extra communication overhead, unlike the non-privacy preserving version. The secure dot product operation has three times the communication cost of a regular two-party dot operation. It is also of $\mathcal{O}(n)$ linear complexity with the number of overlapping users. As the parties each hold half of the user latent factors, they must interact to complete the dot product operation. This communication is also of size $\mathcal{O}(n)$ with the number of overlapping users. Additionally, the communication can be made linear for discovering the user overlap, as exemplified by [11].

Finally, the underlying privacy-preserving matrix factorization approach also incurs communication costs, as discussed in Chapter 4.3.3.

5.4.4. Privacy Analysis

For the privacy requirements, we address the item and rating privacy by using the additive separation approach for privacy-preserving matrix factorization. We present the privacy analysis of this approach in Chapter 4.3.4. Briefly, the additive separation approach provides item privacy due to the pre-image resistance property of the keyed hash function. On the other hand, the rating privacy is related to the ϵ parameter, with the increased noise level corresponding to higher errors for adversarial parties trying to obtain the ratings in non-colluding settings. We also note that the rating privacy is lost when parties collude.

We address the third requirement through the private set intersection protocols that ensure the identifiers of non-overlapping elements of the sets are not revealed to either of the parties.

Our design meets the fourth requirement: using only the user latent factors and keeping the other model outputs private within the domains for the similarity calculation. However, leaking the user latent factors still poses a privacy threat as paired with the rating distribution or range information, the other model outputs can be brute-forced. Our design incorporates secure dot product operations to calculate the inter-domain and intra-domain similarities to prevent user latent factor sharing. The secure dot product operations reveal the dot product of the vectors and not the vectors themselves to the involved parties. Therefore, the vectors can not be recovered from the dot products, as the number of known equations is two: the dot product of two vectors, one known to the target domain and the self-product of the unknown vector from the source domain. When the dimension of the vector is greater than two, the number of unknowns will be greater than the number of equations, thus rendering it impossible for the target domain to reveal the exact vectors of the source domain after the similarity calculation.

6 Conclusion

Recommender systems are ubiquitous in our lives, especially with the recent increase in interaction and feedback mechanisms. In the healthcare setting, patients rate their physicians and leave reviews as a way to express their personal experiences. Online physician ratings are common practice, unfortunately, with little to no consideration for patient privacy. Moreover, new patients require transferring information between domains to receive meaningful recommendations based on their prior experience. This cross-domain task brings forth more unaddressed privacy considerations.

Designing cross-domain recommendation systems is a research area with many publications, none of which explore privacy with efficiency. Through this thesis, we hoped to address this research gap and propose a solution that offers cross-domain functionalities for the new user problem with privacy properties and little to no performance loss. Furthermore, we prioritized designing a solution that can instantly replace existing recommender systems, not compromising runtime or accuracy. In this chapter, we first discuss our results, outline the limitations of our design and propose directions for future research.

6.1. Discussion

To restate our research question:

How can we design an efficient privacy-preserving cross-domain recommender system with partially overlapping users?

In order to answer this question, we start by designing a privacy-preserving matrix factorization scheme, on which we design our cross-domain solution and provide proof-of-concept implementations.

Efficient in terms of runtime performance: We use differential privacy to add a formal definition of privacy while maintaining the regular factorization processes, as the noise added data could be processed the same as the non-privacy settings. Noise addition allowed us to address our research question's ”efficient” requirement, and we demonstrated that the runtime is not significantly affected by the introduced privacy measures. We also performed a communication overhead analysis and proposed some optimizations to reduce further the overhead that stems from adding privacy. Overall, our scheme is efficient in terms of runtime performance compared to other privacy-preserving matrix factorization designs in the literature.

Efficient in terms of predictive performance: To prevent the accuracy loss introduced by the differential privacy noise, we proposed constructing a two-party setting. Our two approaches are: increasing the redundancy of data and the operations or separating the rating into two additive parts masked with noise. We found that the additive separation approach performs best in countering predictive performance loss, followed by the increased redundancy approach with oppositely signed noise through experiments. We demonstrated a clear trade-off between rating privacy and the recommender system's predictive performance in our proposed methods.

Privacy-preserving: Our design allows recommender systems to determine the level of privacy they require with respect to the level of predictive performance loss they can tolerate. We demonstrate the level of rating privacy through the experiments of adversaries guessing the ratings and measuring the error. For item privacy, we use a keyed hash function over the physician identifiers, hiding the physician information associated with the rating against recommender systems and potential eavesdroppers.

Cross-domain recommender system with partially overlapping users: For this functionality, we first design a basis cross-domain recommender system for the new user case. Then we add privacy elements using the privacy-preserving matrix factorization, private set intersection for discovering the user overlap, and secure dot product operations for user similarity calculation. Based on our proof of concept implementation, we conclude that with over 10% user overlap, the predictive performance increases with increased user overlap. We also demonstrate a similar privacy and accuracy trade-off and that predictive performance loss is minor in this construction with respect to the added privacy noise. Finally, we show that the cost of adding privacy to our cross-domain recommender system is negligible regarding predictive performance loss, runtime cost, and communication overhead.

6.2. Limitations

We employ some assumptions and simplifications in our design to present a solution. In this section, we will discuss the limitations that stem from our assumptions.

First and foremost, our central assumption is the users' reliability in the rating scheme. We assumed users' ratings are in a predetermined range according to the application domain requirements. In online physician rating systems, generally, this range is set by the domain provider. As the users submit their rating openly through the websites such as [13, 30, 8], it is trivial to validate that the rating is in a given range. However, in our system, the recommendation providers receive noisy ratings. The added noise causes the submitted rating to be far beyond the given range, and the recommender system cannot verify that the exact rating masked by the noise is of an acceptable range. However, our design allows for discerning repeat rating submissions from one user profile. The healthcare domain prevents users from registering multiple times, thus preventing shilling attacks [40]. Overall, our design is not suitable for cases of unreliable users who submit invalid ratings.

Additionally, we safeguard item privacy using a keyed hash function. The key is shared among all system users while being secret to the recommender systems. This assumption, in reality, is difficult to keep, as it means even one user collaborating with the recommender system can reveal all of the associated items. In case of a leak, refreshing the key requires updating all item mappings. We do not outline a formal mechanism for possible key refreshment and assume no patient leaks the key. Similar assumptions of non-collaboration are present in our work, such as between the domains, which is acceptable for our semi-honest setting but would not hold for adversarial situations.

Finally, we make a simplification while evaluating the adversary who is trying to guess the exact ratings. In reality, the parties aware that the masked ratings they receive are masked with only positive or negative noise values might behave differently than what we propose. For example, they can generate oppositely signed noise to discern the underlying rating better or project the ratings to the normal range with methods other than clipping.Therefore, our adversarial implementation in this work should be reviewed with a grain of salt.

6.3. Future Work

Following our contributions to the design of a privacy-preserving cross-domain recommender system, we note that there are still open areas for further research:

Adding privacy to user similarities The privacy requirements for our system allow for user similarities to be public information as neither the source nor the target domain possesses rating information. Therefore similarities between the users can not be used to deduce rating patterns of individual users. However, real-life scenarios might allow domains to possess extra auxiliary information, and the user similarities can be leveraged to hurt user privacy. Therefore, a direction for future work is focusing on protecting the user similarities with differential privacy or secret sharing schemes.

Improving recommendations on existing overlapping users Our system focuses on generating recommendations for the cold start users without interaction in the target domain. However, existing recommender system research focuses on also improving recommendation performance on existing users with some interaction. In addition, domain adaptation strategies leverage target domain ratings to generate recommendations for overlapping but non-cold start users, such as using a change-of-basis method over latent factors or finding transformation matrix [22]. Future work can extend our design also to support existing users.

Extending the scheme to be dual-target Dual-target cross-domain recommendations are defined as systems where not only the target domain but the source domain recommendations are enhanced[41]. Our system is single-target, with only the target domain recommendations generated. However, we already calculate the inter-domain similarities in our scheme, which can be extended to the source domain for dual-target purposes.

Testing on a variety of datasets We present accuracy and privacy trade-off in our system and suggest that service providers determine their preferred level of privacy with respect to the predictive performance loss. Future work can include other datasets such as MovieLens 25M, Amazon Review dataset, and a synthetic dataset, which would inform recommendation providers better. Furthermore, public websites can be scraped to generate a robust physician rating dataset to reflect the physician rating behavior fully.

Testing the accuracy and privacy trade-off for a greater number of parties In our work, we present the design with a two-party setting and analyze the trade-off. As we hypothesize, the two-party setting allows for the sensitivity to be halved and the same level of privacy to be achieved while maintaining accuracy. Further research can determine how the increased number of parties affects our proposed trade-off.

6.4. Closing Remarks

Physician recommender systems will only be more critical as patients interact with healthcare systems and require more assistance with finding the most appropriate provider. In this work, we demonstrated that this pursuit need not come with the cost of patient privacy. Instead, we provide an efficient design that selects the desired privacy level, is easily adoptable by existing systems and can generate meaningful recommendations for patients without interactions in a new setting, based on their previous experience with physicians. While more work is still on the horizon for privacy-preserving cross-domain recommendation systems, we believe our design, implementation, and evaluation are a step in the right direction, providing provable privacy for patients requiring guidance.

References

[1] Deepak Agarwal, Bee-Chung Chen, and Bo Long. “Localized factor models for multi-context recommendation”. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*. Ed. by Chid Apté, Joydeep Ghosh, and Padhraic Smyth. ACM, 2011, pp. 609–617. DOI: 10.1145/2020408.2020504. URL: https://doi.org/10.1145/2020408.2020504.

[2] Donald Beaver. “Commodity-Based Cryptography (Extended Abstract)”. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. Ed. by Frank Thomson Leighton and Peter W. Shor. ACM, 1997, pp. 446–455. DOI: 10.1145/258533.258637. URL: https://doi.org/10.1145/258533.258637.

[3] Arnaud Berlioz et al. “Applying Differential Privacy to Matrix Factorization”. In: *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*. Ed. by Hannes Werthner et al. ACM, 2015, pp. 107–114. DOI: 10.1145/2792838.2800173. URL: https://doi.org/10.1145/2792838.2800173.

[4] Iván Cantador et al. “Cross-Domain Recommender Systems”. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer, 2015, pp. 919–959. DOI: 10.1007/978-1-4899-7637-6_27. URL: https://doi.org/10.1007/978-1-4899-7637-6%5C_27.

[5] Fran Casino and Constantinos Patsakis. “An Efficient Blockchain-Based Privacy-Preserving Collaborative Filtering Architecture”. In: *IEEE Trans. Engineering Management* 67.4 (2020), pp. 1501–1513. DOI: 10.1109/TEM.2019.2944279. URL: https://doi.org/10.1109/TEM.2019.2944279.

[6] Chaochao Chen et al. “Differential Private Knowledge Transfer for Privacy-Preserving Cross-Domain Recommendation”. In: *WWW ’22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. Ed. by Frédérique Laforest et al. ACM, 2022, pp. 1455–1465. DOI: 10.1145/3485447.3512192. URL: https://doi.org/10.1145/3485447.3512192.

[7] Emiliano De Cristofaro and Gene Tsudik. “Practical Private Set Intersection Protocols with Linear Complexity”. In: *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers*. Ed. by Radu Sion. Vol. 6052. Lecture Notes in Computer Science. Springer, 2010, pp. 143–159. DOI: 10.1007/978-3-642-14577-3_13. URL: https://doi.org/10.1007/978-3-642-14577-3%5C_13.

[8] Docplanner Group. *Making the healthcare experience more human*. 2022. URL: https://www.docplanner.com/about-us (visited on 02/28/2022).

[9] Wenliang Du and Zhijun Zhan. “Building Decision Tree Classifier on Private Data”. In: *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*. CRPIT ’14. Maebashi City, Japan: Australian Computer Society, Inc., 2002, pp. 1–8. ISBN: 0909925925.

[10] Cynthia Dwork et al. “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 265–284. DOI: `10.1007/11681878\_14`. URL: `https://doi.org/10.1007/11681878%5C_14`.

[11] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. “Private Set Intersection with Linear Communication from General Assumptions”. In: *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, WPES@CCS 2019, London, UK, November 11, 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, and Josep Domingo-Ferrer. ACM, 2019, pp. 14–25. DOI: `10.1145/3338498.3358645`. URL: `https://doi.org/10.1145/3338498.3358645`.

[12] Simon Funk. *Netflix update: Try this at home*. 2006. URL: `https://sifter.org/simon/journal/20061211.html`.

[13] Hangzhou WeDoctor Health Technology Co., Ltd. *About us*. 2022. URL: `https://www.guahao.com/about` (visited on 02/28/2022).

[14] Liangliang He et al. “LFSF: Latent Factor-Based Similarity Framework and Its Application for Collaborative Recommendation”. In: *Advances in Multimedia Information Processing - PCM 2018 - 19th Pacific-Rim Conference on Multimedia, Hefei, China, September 21-22, 2018, Proceedings, Part I*. Ed. by Richang Hong et al. Vol. 11164. Lecture Notes in Computer Science. Springer, 2018, pp. 744–754. DOI: `10.1007/978-3-030-00776-8\_68`. URL: `https://doi.org/10.1007/978-3-030-00776-8%5C_68`.

[15] T. Ryan Hoens et al. “Reliable medical recommendation systems with patient privacy”. In: *ACM Trans. Intell. Syst. Technol.* 4.4 (2013), 67:1–67:31. DOI: `10.1145/2508037.2508048`. URL: `https://doi.org/10.1145/2508037.2508048`.

[16] Zhaoyan Hu et al. “A novel privacy-preserving matrix factorization recommendation system based on random perturbation”. In: *J. Intell. Fuzzy Syst.* 38.4 (2020), pp. 4525–4535. DOI: `10.3233/JIFS-191287`. URL: `https://doi.org/10.3233/JIFS-191287`.

[17] Stefan Katzenbeisser and Milan Petkovic. “Privacy-Preserving Recommendation Systems for Consumer Healthcare Services”. In: *Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008, March 4-7, 2008, Technical University of Catalonia, Barcelona , Spain*. IEEE Computer Society, 2008, pp. 889–895. DOI: `10.1109/ARES.2008.85`. URL: `https://doi.org/10.1109/ARES.2008.85`.

[18] Harmanjeet Kaur, Neeraj Kumar, and Shalini Batra. “An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system”. In: *Future Gener. Comput. Syst.* 86 (2018), pp. 297–307. DOI: `10.1016/j.future.2018.03.017`. URL: `https://doi.org/10.1016/j.future.2018.03.017`.

[19] Muhammad Murad Khan, Roliana Ibrahim, and Imran Ghani. “Cross Domain Recommender Systems: A Systematic Literature Review”. In: *ACM Comput. Surv.* 50.3 (2017), 36:1–36:34. DOI: `10.1145/3073565`. URL: `https://doi.org/10.1145/3073565`.

[20] Jinsu Kim et al. “Efficient Privacy-Preserving Matrix Factorization for Recommendation via Fully Homomorphic Encryption”. In: *ACM Trans. Priv. Secur.* 21.4 (2018), 17:1–17:30. DOI: `10.1145/3212509`. URL: `https://doi.org/10.1145/3212509`.

[21] Yehuda Koren, Robert M. Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems”. In: *Computer* 42.8 (2009), pp. 30–37. DOI: `10.1109/MC.2009.263`. URL: `https://doi.org/10.1109/MC.2009.263`.

[22] Tong Man et al. “Cross-Domain Recommendation: An Embedding and Mapping Approach”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. Ed. by Carles Sierra. ijcai.org, 2017, pp. 2464–2470. DOI: 10.24963/ijcai.2017/343. URL: https://doi.org/10.24963/ijcai.2017/343.

[23] Kartik Nayak et al. “GraphSC: Parallel Secure Computation Made Easy”. In: *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 377–394. DOI: 10.1109/SP.2015.30. URL: https://doi.org/10.1109/SP.2015.30.

[24] Valeria Nikolaenko et al. “Privacy-preserving matrix factorization”. In: *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM, 2013, pp. 801–812. DOI: 10.1145/2508859.2516751. URL: https://doi.org/10.1145/2508859.2516751.

[25] Taiwo Blessing Ogunseyi, Cossi Blaise Avoussoukpo, and Yiqiang Jiang. “Privacy-Preserving Matrix Factorization for Cross-Domain Recommendation”. In: *IEEE Access* 9 (2021), pp. 91027–91037. DOI: 10.1109/ACCESS.2021.3091426. URL: https://doi.org/10.1109/ACCESS.2021.3091426.

[26] Taiwo Blessing Ogunseyi, Tang Bo, and Cheng Yang. “A privacy-preserving framework for cross-domain recommender systems”. In: *Comput. Electr. Eng.* 93 (2021), p. 107213. DOI: 10.1016/j.compeleceng.2021.107213. URL: https://doi.org/10.1016/j.compeleceng.2021.107213.

[27] Weike Pan et al. “Transfer Learning in Collaborative Filtering for Sparsity Reduction”. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. Ed. by Maria Fox and David Poole. AAAI Press, 2010. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1649.

[28] Ashish K. Sahu and Pragya Dwivedi. “Matrix factorization in Cross-domain Recommendations Framework by Shared Users Latent Factors”. In: *Procedia Computer Science* 143 (2018), pp. 387–394. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.10.410. URL: https://www.sciencedirect.com/science/article/pii/S1877050918321070.

[29] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer, 2016. ISBN: 978-3-319-21935-6. DOI: 10.1007/978-3-319-21936-3. URL: https://doi.org/10.1007/978-3-319-21936-3.

[30] *The Original Doctor Rating Review Site*. https://www.ratemds.com/about/.

[31] Thi Ngoc Trang Tran et al. “Recommender systems in the healthcare domain: state-of-the-art and research issues”. In: *J. Intell. Inf. Syst.* 57.1 (2021), pp. 171–201. DOI: 10.1007/s10844-020-00633-6. URL: https://doi.org/10.1007/s10844-020-00633-6.

[32] André Calero Valdez and Martina Ziefle. “The users’ perspective on the privacy-utility trade-offs in health recommender systems”. In: *Int. J. Hum. Comput. Stud.* 121 (2019), pp. 108–121. DOI: 10.1016/j.ijhcs.2018.04.003. URL: https://doi.org/10.1016/j.ijhcs.2018.04.003.

[33] Xinghua Wang et al. “CDLFM: cross-domain recommendation for cold-start users via latent feature mapping”. In: *Knowl. Inf. Syst.* 62.5 (2020), pp. 1723–1750. DOI: 10.1007/s10115-019-01396-5. URL: https://doi.org/10.1007/s10115-019-01396-5.

[34] Saul J Weiner et al. “Patient-Centered Decision Making and Health Care Outcomes: An Observational Study”. In: *Annals of Internal Medicine* 158.8 (2013), p. 573. ISSN: 0003-4819. DOI: 10.7326/0003-4819-158-8-201304160-00001.

[35] Udi Weinsberg et al. “BlurMe: inferring and obfuscating user gender based on ratings”. In: *Sixth ACM Conference on Recommender Systems, RecSys ’12, Dublin, Ireland, September 9-13, 2012*. Ed. by Padraig Cunningham et al. ACM, 2012, pp. 195–202. DOI: 10.1145/2365952.2365989. URL: https://doi.org/10.1145/2365952.2365989.

[36] Zhenzhen Xu et al. “Cross-domain item recommendation based on user similarity”. In: *Comput. Sci. Inf. Syst.* 13.2 (2016), pp. 359–373. DOI: 10.2298/CSIS150730007Z. URL: https://doi.org/10.2298/CSIS150730007Z.

[37] Xu Yu et al. “A Privacy-Preserving Cross-Domain Healthcare Wearables Recommendation Algorithm Based on Domain-Dependent and Domain-Independent Feature Fusion”. In: *IEEE J. Biomed. Health Informatics* 26.5 (2022), pp. 1928–1936. DOI: 10.1109/JBHI.2021.3069629. URL: https://doi.org/10.1109/JBHI.2021.3069629.

[38] Mingwu Zhang, Yu Chen, and Jingqiang Lin. “A Privacy-Preserving Optimization of Neighborhood-Based Recommendation for Medical-Aided Diagnosis and Treatment”. In: *IEEE Internet Things J.* 8.13 (2021), pp. 10830–10842. DOI: 10.1109/JIOT.2021.3051060. URL: https://doi.org/10.1109/JIOT.2021.3051060.

[39] Qian Zhang et al. “A Cross-Domain Recommender System With Kernel-Induced Knowledge Transfer for Overlapping Entities”. In: *IEEE Trans. Neural Networks Learn. Syst.* 30.7 (2019), pp. 1998–2012. DOI: 10.1109/TNNLS.2018.2875144. URL: https://doi.org/10.1109/TNNLS.2018.2875144.

[40] Wei Zhou et al. “Shilling attack detection for recommender systems based on credibility of group users and rating time series”. In: *PLOS ONE* 13.5 (May 2018), pp. 1–17. DOI: 10.1371/journal.pone.0196533. URL: https://doi.org/10.1371/journal.pone.0196533.

[41] Feng Zhu et al. “Cross-Domain Recommendation: Challenges, Progress, and Prospects”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. Ed. by Zhi-Hua Zhou. ijcai.org, 2021, pp. 4721–4728. DOI: 10.24963/ijcai.2021/639. URL: https://doi.org/10.24963/ijcai.2021/639.